Escuela Politécnica Superior

Desarrollo de un Videojuego en Unity: To the Heaven

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Juan Rubio Navarro

Tutor/es:
Carlos Villagrá Arnedo

Diciembre 2020

Universitat d'Alacant
Universidad de Alicante

AGRADECIMIENTOS

A mi familia, por ayudarme y aconsejarme en mis decisiones. A mis amigos, especialmente aquellos a los que he conocido en la carrera, por haber hecho esta etapa tan feliz e inolvidable. A los profesores, por enseñarme mucho de los que se A Carlos por ser mi tutor y animarme a seguir adelante con esta propuesta.

Dar las gracias también a Patrick de Arteaga por los recursos sonoros y a Mark Turnbull por algunos recursos gráficos.

CITAS

“El tiempo lo cambia todo ... eso es lo que la gente dice, pero no es verdad. Hacer cosas cambia las cosas. No hacer nada deja las cosas exactamente como están”. – Dr House.

“Todos tomamos decisiones en la vida, pero al final son las decisiones las que nos moldean”. Andrew Ryan

“Las personas son rápidas para juzgar, pero lentas para corregirse a sí mismas”. Ezio Auditore.

Índice de Contenidos

Índice de figuras

Índice de Tablas

1. Introducción

El auge de los videojuegos es imparable, la industria crece año tras año y cada vez atrae a más y más gente, los cuales disfrutan de esta maravillosa industria. Gracias a ello, muchos de ellos, de una forma u otra, quieren formar parte de ella de una forma u otra.

Vivimos la época dorada de los videojuegos, tenemos acceso a muy bajo precio o incluso de forma gratuita a cientos de videojuegos. Las plataformas han evolucionado y se han acercado al consumidor, nunca había sido tan fácil consumir videojuegos y aún más importante, nunca había sido tan accesible desarrollar videojuegos.

Crear un videojuego es una tarea compleja, ser requieren muchos conocimientos, práctica, dedicación y pasión por parte de sus desarrolladores. Actualmente el mercado goza de una masa de usuarios enorme, la cantidad de títulos que se lanzan, sumando al bajo precio de los títulos y la competitividad hacen que sea un mercado muy atractivo y lleno de oportunidades, pero en el que es muy complicado destacar.

El proyecto se centra en los videojuegos para dispositivos móviles, ya que esta será nuestra plataforma objetivo y el género de puzles el cual será el principal del videojuego. El proyecto será desarrollado en C# utilizando el motor Unity 3D.

En esta memoria, se pretende documentar el desarrollo de una versión demo de un videojuego, pero dándole especial importancia, más incluso que al producto en sí, en documentar todos los pasos del desarrollo que sufre un videojuego, que pretender ser lanzado. Haciendo todos los estudios previos necesarios, realizando una planificación del proyecto entero y trabajando mediante metodologías ágiles, estudiando la viabilidad de este, el software necesario, los aspectos legales para su publicación, referencias creativas y mucho más para dotar al documento de la profundidad necesaria.

Para ello, en primer lugar, se contextualizará brevemente la situación actual de la industria y poco a poco se irá entrando en detalle en todos los conceptos necesarios hasta llegar al momento del desarrollo donde se mostrarán como se aplican los conocimientos previos y lo diseñado el GDD, que se realizará de forma previa.

2.Marco teórico

En este bloque todo lo relacionado con el mundo de los videojuegos, comenzando a focalizarnos en el proyecto que nos atañe. En primer lugar, se hablará sobre los videojuegos en general y la industria, el género del juego en cuestión y las referencias principales del género.

2.1 Concepto de Videojuego

No hay una única definición que englobe todas las características que tienen los videojuegos, por lo que para dar una es necesario generalizar y primar los elementos más comunes dentro de las infinitas variables que existe.

Una posible definición, teniendo en cuenta lo anterior podría ser: *Conjunto de elementos multimedia diseñados para, en conjunto, entretener/divertir al usuario mediante la interacción directa con el mismo, a través de una pantalla y algún sistema de control.* (Julián Pérez Porto, 2010).

Esta conceptualización propia surge de mi propia experiencia y de la investigación para dar forma a una definición que abarque la mayor cantidad de videojuegos posible

2.2 La industria de los videojuegos

Los videojuegos son una industria que crece a pasos agigantados cada año, es la industria del siglo, el crecimiento que está experimentando estos años tanto a nivel económico como a nivel social es espectacular, se ha pasado de una industria de nicho, donde a aquellas personas que jugaban a videojuegos se les llegaba incluso a discriminar, a una industria que con la primera PlayStation de Sony llegó a nuestras casas de forma masiva. Posteriormente con los dispositivos móviles, que han invadido nuestra vida y les dedicamos horas cada día, han llegado a nuestro bolsillo y nos acompañan en nuestro día a día.

Desde el primer videojuego de la historia "*Tenis For Two*" creado en 1958 hasta ahora se han creado millones de videojuegos, desde pequeños títulos creados por una sola persona a lo largo de muchos años en sus ratos libres a superproducciones millonarias donde trabajan miles de personas con unos resultados técnicos simplemente espectaculares. Siendo ambas alternativas perfectamente viables, pues se ha demostrado que tanto las superproducciones (*Juegos AAA*) como los juegos pequeños (*Juegos Indie*) han dado rédito económico y fama en el sector a sus creadores. Debido a que no existe una forma real de clasificar un videojuego

entre *Indie o AAA*, no se tendrá en cuenta esta distinción a la hora de hablar de datos, aunque sería algo realmente interesante.

Figura 1- Evolución Ingresos por plataformas, 2018

Fuente: https://perfilindustrial.com/la-industria-de-videojuegos-concentra-cada-vez-mas-el-interes-de-extranjeros/

En la Figura 1 se puede observar claramente la evolución de la industria en los últimos años. En primer lugar, se ve un incremento anual medio, aproximadamente, de un 10% en los ingresos totales que genera la industria, lo que muestra claramente que se aún se encuentra en plena expansión generando unas cantidades de dinero increíbles. El otro factor que destacar se encuentra en la distribución por plataforma. Se observa como el mercado móvil ha experimentado una gran masificación, gracias a los smartphones, multiplicando por 3 su presencia y con previsiones de ser el dominador absoluto del sector. En la imagen Figura 2 se encuentra la explicación de este aumento.

2.2.1 Expansión del mercado móvil

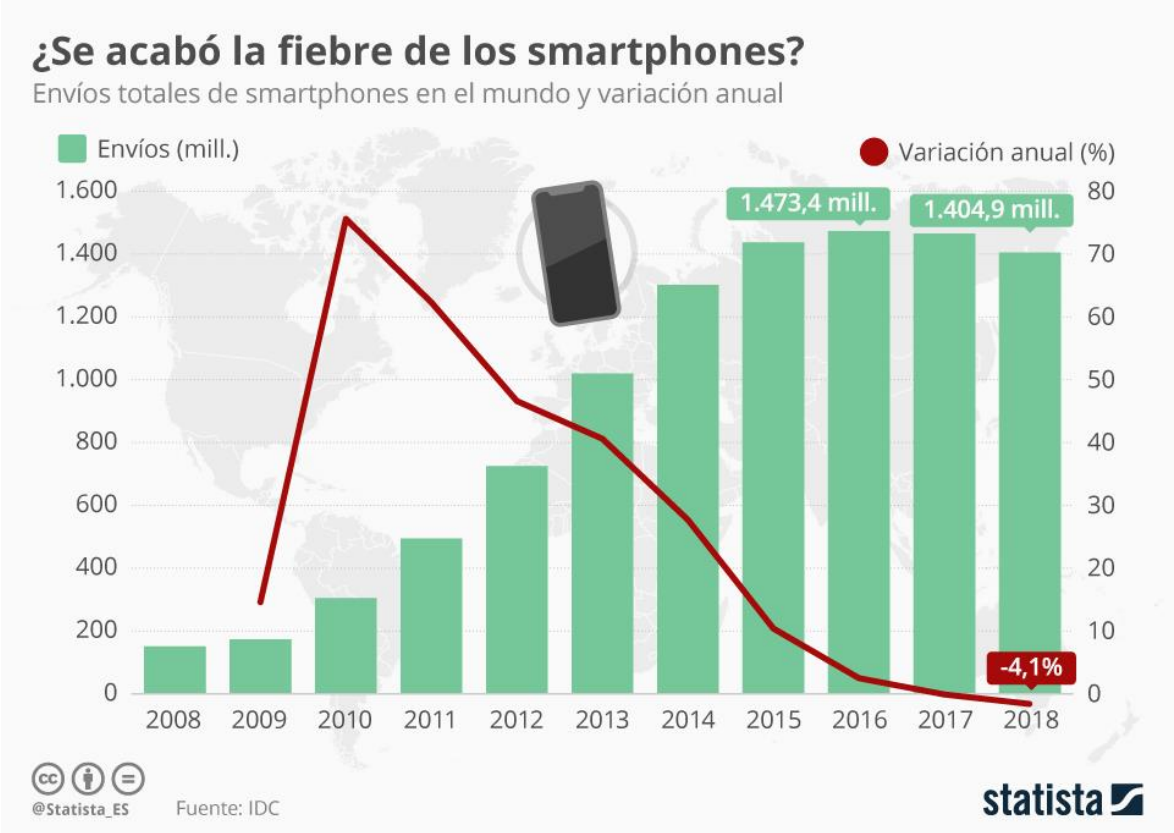


Figura 2- Evolución Ventas de Smartphones, 2018

Fuente https://es.statista.com/grafico/17040/envios-totales-de-smartphones-en-el-mundo/

Como se puede observar a simple vista, las gráficas de las Figuras 1-2 están muy relacionadas. En los últimos diez años las ventas de smartphones se han octuplicado, alcanzando una expansión mundial inalcanzable para cualquier otra videoconsola o el pc, pese a que las ventas estén comenzando ya a disminuir, por ejemplo, la suma de las consolas de sobremesa más vendidas, Ps4, Xbox y Nintendo Switch no alcanza los doscientos millones. Al llegar a tanta gente y disponer de infinidad de productos a bajo precio o gratuitos se entiende la expansión y se puede establecer una correlación directa entre el auge del mercado móvil y del éxito de los videojuegos para estos.

2.2.2 La industria de los videojuegos vs el resto de las industrias.

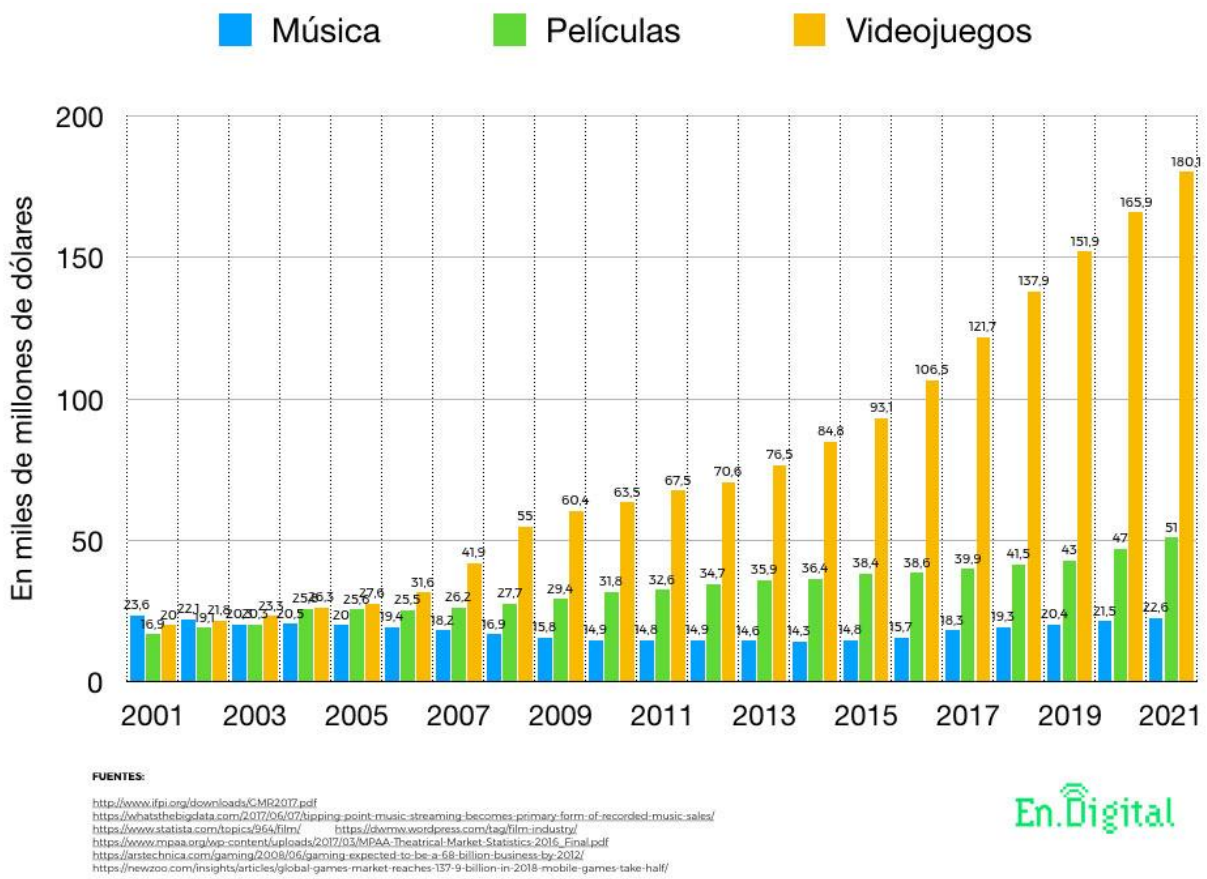


Figura 3– Comparación ingresos de las principales industrias mundiales

Fuente https://es.statista.com/grafico/17040/envios-totales-de-smartphones-en-el-mundo/

En la Figura 3 se puede observar que la industria de los videojuegos, desde hace años, supera tanto al cine cómo a la música por mucho, debido a su muy superior crecimiento anual y a la popularización de estos, se prevé que estos sigan así durante varios años y no hay estimaciones de cuando se alcanzará el máximo.

2.2.3 Conclusiones sobre la industria

En resumen, los videojuegos, a nivel industrial, gozan de una situación única gracias a, principalmente, el auge los dispositivos móviles y a la popularización del sector. Esta situación es esperable, vistas las tendencias, que se mantenga durante bastantes años por lo que, como se viene diciendo hace un tiempo, se puede decir que vivimos *"la edad de oro"* de los videojuegos.

2.3 La figura del desarrollador

El sector ha experimentado, fruto de su expansión, una gran profesionalización. Los perfiles necesarios para crear un videojuego se han multiplicado y diversificado. Millones de personas en el mundo se dedican a este sector, siendo algunos de los perfiles profesionales más relevantes: programadores, modeladores, músicos, artistas, guionistas… Cada una se encarga de una labor específica para en conjunto crear un título, especialmente en el desarrollo *AAA*, dando como resultado grandes títulos y en algunos casos auténticas obras de arte.

Esta profesionalización ha supuesto la creación de una gran cantidad de contenido formativo como: cursos, formación profesional, grados y másteres. Las instituciones se dan cuenta que es una industria muy importante en la actualidad, que en el futuro será vital y necesitan gente que se forme en estos ámbitos.

2.4 Definición e Historia de los videojuegos de puzles

Un videojuego de puzles o lógica es según una definición clásica es "aquel que plantea al jugador una serie de retos que requieren de algún tipo de ejercicio mental por parte de los usuarios para ser resueltos". Esta definición, aunque es correcta y perfectamente válida creo que olvida una parte fundamental en los videojuegos, el entretenimiento del jugador, por ello creo que la definición de Scott Kim, famoso diseñador de puzles, la complementa de buena forma. "Los puzles son problemas divertidos, una especie de reto que en su resolución (no importando lo compleja o abstracta que ésta pueda ser) entretiene al usuario" (Atomix, 2013).

Desde que aparecieron los videojuegos los puzles han estado presentes en la historia, "de hecho, la creación de ellos no tardó en convertirse en un ejercicio común de programación y diseño de juegos electrónicos". Ya en los 70 aparecieron los primeros videojuegos de puzles, pero no fue hasta los 80 cuando los puzles alcanzaron fama mundial gracias a los videojuegos con el lanzamiento de *Tetris* (Atari, 2000). Desde ese momento, hasta nuestros días, los puzles han ido de la mano de los videojuegos y se han vuelto inseparables.

Es un género que, aunque ha dados títulos centrados exclusivamente en la resolución de retos, es más conocido por ser un subgénero capa de ser introducido en prácticamente cualquier videojuego, independientemente del género principal de este, su plataforma o

cualquier otro factor. Esto se debe a la gran cantidad de variantes que existen, a su flexibilidad y al usual consumo por parte de la sociedad de este tipo de contenido.

"Uno de los grandes atractivos de los puzles es el contraste entre lo sencillo y lo complejo: mover piezas, agrupar colores, rotar una figura y acomodarla en cierto espacio, etc." (Atomix, 2013)

2.5 Tipos de Puzles

Como ya se ha comentado, el género tiene una gran cantidad de variantes muy diferentes entre sí, debido a la inmensidad de tipos de puzles se destacarán solo los tipos más recurrentes y conocidos en los videojuegos agrupándolos en bloques (Calveley, 2015).

2.5.1 Uso de objetos

Son los puzles más sencillos, consisten en la necesidad de utilizar un objeto para poder completar el nivel u obtener objetos que faciliten el progreso

Una saga que utiliza desde hace muchos años este sistema es *Resident Evil*, donde el jugador debe recolectar recursos por el escenario que pueden ser combinador desde el menú que aparece en la Figura 5 para obtener nuevos objetos con nuevas funcionalidades.

Figura 4- Resident Evil 7, Inventario

Fuente: game.capcom.com/manual/re7/

Estos tipos de puzles suelen emplearse en muchos tipos de videojuegos como una mecánica secundaria básica, normalmente como sistema de *crafteo(creación de objetos mediante un menú)*, son sistemas que no requieren prácticamente de ninguna instrucción o tutorial debido a su simpleza, pero sobre todo se debe a que los jugadores están muy acostumbrados a estos sistemas.

2.5.2 Orden y Navegación

Son aquellos puzles que consisten en realizar una serie de pasos una en una secuencia determinada u ordenar una serie de elementos. Hay muchas variantes como pulsar una serie de botones/palancas en un orden concreto, desplazar fichas sobre una cuadrícula para formar una imagen o recorrer un camino sobre una zona donde el suelo cae si pisas en el sitio equivocado.

En este caso la saga *Resident Evil* vuelve a ser un buen ejemplo, en este caso el jugador debe interactuar con una serie de paneles y resolver los circuitos para poder avanzar.

Figura 5- Resident Evil 2 remake, Puzle de panel eléctrico

Fuente: Meristation.com

Este tipo de puzles son muy comunes y llevan utilizándose muchos años, son diseñados como minijuegos en los que el jugador debe en primer lugar interactuar con algún objeto como un ordenador o panel de electricidad para acceder a una interfaz especial en la que se desarrollará el puzle. Cada puzle es distinto y obedecerá únicamente a las limitaciones que hayan establecidos los desarrolladores. Es común que estos minijuegos cuenten con restricciones de algún tipo, como un límite de movimientos o una cuenta atrás para añadirle un poco de dificultad y requerirle al jugador un poco de agilidad mental extra y reflejos.

2.5.3 Conversacionales

Consisten en entablar un diálogo con un *NPC*, que se desarrolla a partir de una serie de opciones que se le ofrecen al jugador.

Esta serie de diálogos son muy característicos de los juegos en los juegos *RPG (juego de rol)* y los *Open World(juegos de mundo abierto)*, géneros en los que es muy común tener muchos diálogos de esta clase.

Unos de los grandes exponentes de esta mecánica y que abarca los dos géneros mencionados es la saga *Fallout*, especialmente en sus primeras entregas.

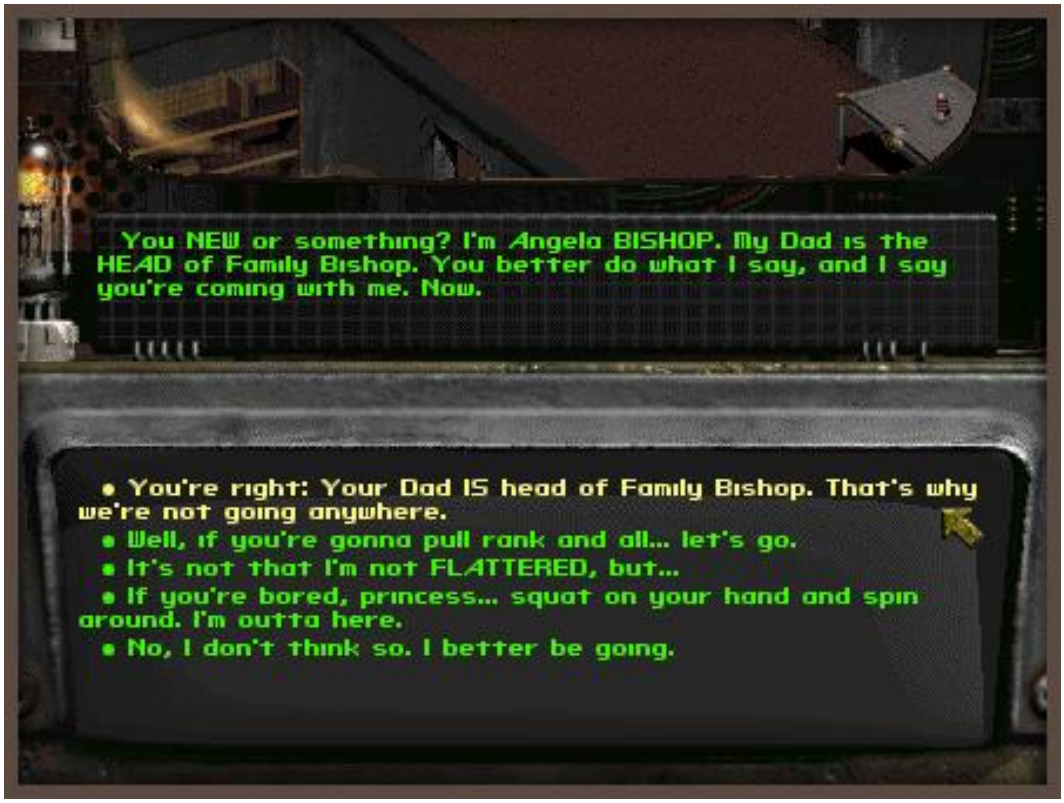


Figura 6- Fallout 2 Negociación con un NPC

Fuente: https://www.mundogamers.com/

Los diálogos son un elemento extremadamente común en los videojuegos, sirven para guiarnos por el videojuego y darnos información sobre los personajes o el entorno del videojuego, sin embargo, este tipo de conversaciones, como ya se ha mencionado, tienen otro objetivo. Están planteadas como un minijuego, el jugador debe deducir en función de lo que le dice el *NPC*, de sus opciones de diálogo y de lo que el mismo quiere conseguir, cual es la mejor respuesta. Para ellos el jugador, normalmente, de forma previa el encuentro tiene o puede tener acceso a información que le de pistas sobre cuál es el comportamiento más adecuado a tener con el *NPC* para conseguir su objetivo.

2.5.4 Precisión

Son un tipo de puzles especiales basados en que el jugador debe realizar algún tipo de acción en un momento exacto para tener éxito. La dificultad en este tipo de acciones es muy variable pudiendo ser fáciles o prácticamente imposibles. Algunos de los más comunes de este tipo de puzles son minijuegos como abrir cerraduras o hackear ordenadores, como sucede en la saga *Fallout*, entrar en sigilo en una zona esquivando la visión de un *NPC* cuando este se gira es otra variante muy común.

Figura 7- Skyrim Minijuego cerradura

Fuente: https://elderscrolls.fandom.com/wiki/The_Elder_Scrolls_Wiki

2.6 Juegos de Referencia

Los juegos de puzles siempre han gozada de fama y es un clásico de los videojuegos, hay infinidad de juegos de este tipo y muchos de ellos han cosechado enormes éxitos, tanto de crítica como en ventas. A continuación, se encuentran los principales juegos que se han utilizado como referencia para la realización de este trabajo: *Monument Valley*, *Nekorama y Dream Machine.*

- **Monument Valley** (Ustwo Games, 2014)Se trata de un videojuego de puzles exclusivo para dispositivos móviles desarrollado y publicado por el estudio *Ustwo Games* en el año 2014. Este título será nuestra principal referencia visual y de diseño, ya que destaca en varios puntos muy relevantes que se deben tener en cuenta si nuestro objetivo es desarrollar un juego de puzles para móviles.
    - El objetivo del juego es superar una serie de niveles que se componen de varios puzles sencillos basados en ilusiones ópticas que el jugador debe resolver para avanzar en cada uno.

Figura 8- Monument valley

Fuente: https://www.lavanguardia.com/tecnologia/videojuegos/20151202/30540925538/monument-valley-gratis-iphone-ipad.html

El juego tuvo una enorme repercusión mundial, cabe destacar que el juego se comercializó a un precio de 3,59 €, lo cual agranda aún más su éxito, pues él no ser gratuito y ser bastante corto no eran punto a su favor, pero aun así cautivó al mundo

Monument Valley destaca en muchos apartados, el primero de ellos y que es imposible no destacar es su belleza. El juego utiliza un estilo visual muy especial, utilizando una grandísima paleta de colores, que cambia en cada nivel, esto al combinarlo con efectos visuales muy cuidados genera un aura única y muy especial. Otro de los apartados en los que destaca es su música, esta esta magistralmente pensada y minuciosamente diseñada para transmitir paz y relajación, incita a la contemplación de cada escenario para descubrir los entresijos de cada puzle. Por último, pero no menos importante, se deben destacar su diseño de niveles, que es perfecto, la utilización de las denominadas "*Figuras imposibles"* para crear ilusiones ópticas que deben ser resueltas para poder avanzar en el nivel, y la excelsa precisión del control del personaje y de los elementos del entorno con los que el jugador debe interactuar. Todo esto hacen de Monument Valley una experiencia única recomendada a todos los jugadores.

El juego recibió una secuela en el año 2017, llamada Monument Valley 2. Esta mantuvo el nivel de calidad del original sin incorporar grandes cambios en la fórmula, por lo tanto, no se realizará una descripción más exhaustiva.

Figura 9- Monument valley

Fuente: https://www.xataka.com/videojuegos/monument-valley-y-la-armonia-del-videojuego-en-el-movil

- **Mekorama** (Magni, 2016): Se trata de un videojuego de puzles exclusivo para dispositivos móviles desarrollado y publicado íntegramente por Martin Magni en el año 2016. Este título será nuestra principal referencia a nivel de jugabilidad y mecánicas, ya que esta, aunque siendo simple, está muy bien pensadas.
    - El objetivo del juego es superar una serie de niveles que se componen de varios puzles sencillos basados en ilusiones ópticas que se deben resolver para avanzar en cada uno.

Figura 10- Mekorama

Fuente: http://www.aplicaciones-android.org/mekorama-un-divertido-y-original-puzle-para-android/

Un punto que es obligatorio destacar de este juego es que fue realizado por una única persona, Martin Magni, al realmente impresionante si se tiene en cuenta lo pulido, bien ejecutado y el gran contenido que tiene disponible, sin lugar a duda un trabajo envidiable y con mucho mérito.

El juego tuvo una enorme repercusión mundial, cabe destacar que el juego se comercializo de forma gratuita, alcanzando más de diez millones de descargas el juego gozó y goza de una crítica sublime, los motivos son los siguientes.

Principalmente, esta fama se debe a los siguientes factores: Lo más destacable es, sin lugar a duda, su diseño de niveles es excelente, cuidando mucho los detalles y garantizando una

curva de dificultad justa que incita a seguir jugando. A medida que el jugador avanza los niveles se complican y van apareciendo nuevas mecánicas, además, gracias a la existencia de un editor de mapas propio y una comunidad bastante activa la cantidad de contenido de la que se puede disfrutar es realmente grande.

Figura 11- Mekorama. Niveles creados por los usuarios.

Fuente: https://mekoramaforum.com/

- **Dream Machine** (Game Digits, 2017): Se trata de un videojuego de puzles exclusivo para dispositivos móviles desarrollado y publicado por Game Digits en el año 2017. Este título es una evolución de los que nos mostró Monument Valley, ya que su base es similar, pero añade nuevos tipos de niveles.
    - Al igual que en Monument Valley, el objetivo del juego es superar una serie de niveles que se componen de varios puzles sencillos basados en ilusiones ópticas que el jugador debe resolver para avanzar en cada uno, con la diferencia de que Dream Machine añade niveles de enfrentamiento con jefes y otro tipo de mecánicas.

Figura 12- Dream Machine

Fuente: http://cyborgs.pro/dream-machine-the-game/

Dream Machine, a diferencia de los otros juegos vistos hasta ahora, no fue un éxito mundial, pese a su bajo precio de tan solo 1.09€, hoy, cuatro años después de su lanzamiento ha sido comprado por unas diez mil personas. Un número muy bajo para, en mi opinión, un juego increíblemente completo, pero que en su salida tuvo algunos problemas con los controles y pequeños bugs. Pese a que estos fueron solucionados esto afectó a la percepción inicial del juego y no consiguió explotar. Este hecho nos hace reflexionar sobre la importancia de tener un producto pulido y bien terminado de lanzamiento, porque no es la primera vez que un gran juego no triunfa todo lo que debería o directamente fracasa por tener de lanzamiento algunos fallos que entorpezcan o impidan al usuario disfrutar una experiencia satisfactoria.

Otro de los problemas que tiene Dream Machine es vivir a la sombra del gran éxito de Monument Valley. Los títulos siempre fueron comparados por los usuarios y en casi todos los casos y debido principalmente a lo comentado anteriormente Dream Machine siempre fue el damnificado

Personalmente, Dream Machine es mi juego favorito de esta lista, esto es porque a diferencia de Monument Valley, juego en el que se inspira, además de utilizar puzles basados en ilusiones ópticas añade niveles con mecánicas diferentes y, sobre todo, el cual es su mayor punto de interés y lo que más le aporta es la inclusión de niveles de enfrentamientos contra jefes finales. Estos niveles aportan aire fresco a la experiencia de juegos.

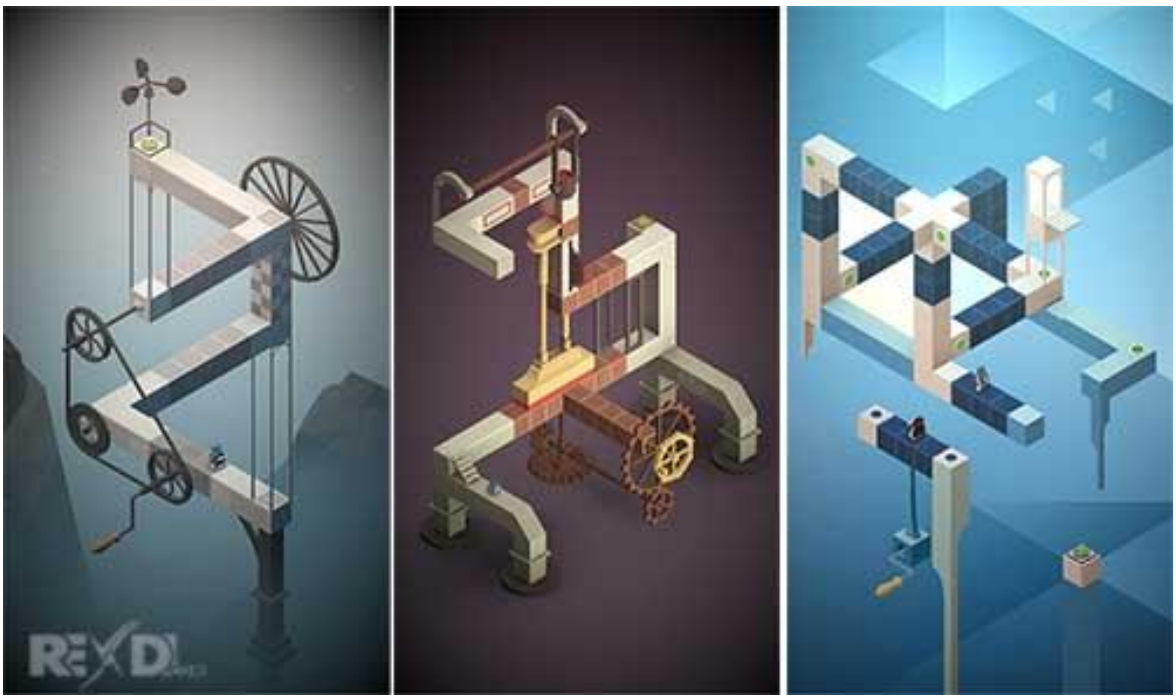


Figura 13 - Dream Machine, niveles

Fuente: https://elandroidelibre.elespanol.com/2016/03/dream-machine-juego-clon-monument-valley.html

2.7.1 Conclusiones sobre el análisis de referencias.

Tras hablar de nuestras referencias (Monument Valley, Mekorama y Dream Machine), es la hora de extraer conclusiones, tanto para quedarnos con los puntos fuertes que nos puedan ayudar con el desarrollo de nuestro videojuego, como con los puntos débiles que hay que tener en cuenta para conseguir realizar un mejor producto. Estos puntos se tratarán en diferentes apartados más adelante.

PUNTOS FUERTES	**PUTNOS DÉBILES**
Control sencillo	Bugs, de cualquier tipo
Niveles cortos	Niveles demasiado largos o complicados
Atractivo visual	Repetitividad
Modelos de negocio realista	Competitividad
Curva de Dificultad	X
Rendimiento	X

Tabla 1– Puntos fuertes y débiles de los juegos analizados

Fuente: Realización propia

3. Justificación y Objetivos

Los videojuegos son mi pasión, desde la primera vez que jugué a un videojuego con cinco años hasta los veintidós, momento en que redacto estas palabras, nunca he dejado de jugar y nunca lo haré, gustándome especialmente los títulos *Indie*, ya que en líneas generales han sido desarrollados por muy pocas personas con otros objetivos más allá de los puramente económicos, un pensamiento con el que simpatizo. Por ellos comencé esta carrera con el sueño de sueño crear videojuegos por mí mismo.

Hasta hace no mucho era imposible que pequeños estudios o personas de forma individual crearan videojuegos ya que las herramientas de desarrollo eran creadas de forma interna por las grandes empresas AAA, pero con la aparición de los motores gráficos gratuitos y otras herramientas esto ya no es así.

El objetivo principal de este TFG, aprovechando la coyuntura actual, es desarrollar una demo de un videojuego con el motor Unity 3D para dispositivos móviles, englobado dentro del género de puzles, un género que gusta mucho y lleno de posibilidades. La idea es desarrollarlo de forma independiente, es decir, utilizando software exclusivamente gratuito que no ponga trabas ni limitaciones al proyecto y creando personalmente todos los elementos necesarios para su desarrollo, haciendo hincapié en la creación de los entornos 3D y exceptuando los elementos sonoros. Otro de los pilares fundamentales del proyecto consistirá en realizar diferentes análisis para conocer de forma concreta el estado actual del mercado y los requisitos para publicar el título una vez se finalice su desarrollo.

Debido a las limitaciones de tiempo de este trabajo, es necesario aclarar que el producto final que se espera obtener es una primera versión jugable con un estilo visual poco utilizado estos días y mecánicas simples. Esto se debe a que el tiempo necesario para desarrollar todo el contenido necesario para poder lanzarlos como producto definitivo es muy superior al disponible.

Teniendo todo lo anterior en cuenta, lo que quiero aprender, expandir mis conocimientos y formarme todo lo posible para que cuando llegue al mundo laboral pueda demostrar que soy Ingeniero Multimedia, capaz de desenvolverse en diversos ámbitos con el objetivo de crear contenido, sean videojuegos o no, con el que hacer disfrutar a los usuarios.

Los objetivos, se podrían resumir en los siguientes:

Relacionado con Unity

- Programación en C#.
- Flujo de trabajo en Unity.
- Programación Móvil.
- Funcionalidades avanzadas y Scripting.
- Programación eficiente.

- Relacionado con el desarrollo de software
  - Fases de análisis, diseño e implementación.
  - Documentación del desarrollo.
  - Investigación.
  - Objetivos y limitaciones
  - Estimación de Costes y riesgos.
  - Modelo de negocio.
  - Planificación.
  - Metodologías de desarrollo.

4. Metodología

Con el objetivo de realizar un desarrollo lo más organizado posible, el cual nos permita avanzar a buen ritmo y de forma eficiente. Para este proyecto se ha optado por el uso de una metodología ágil en lugar de una metodología tradicional. Esta decisión se justifica en las ventajas que aporta esta forma de desarrollar para un proyecto de este tipo, estas están reflejadas en la Tabla 2.

Tradicionales	Ágiles
Resistencia a los cambios	Preparados para cambios
Impuestas por el equipo	Impuestas externamente
Arquitectura esencial, expresada mediante modelos	Menos énfasis en la arquitectura del software
Más roles	Pocos roles
Más artefactos	Pocos artefactos
Grupos grandes y distribuidos	Grupos pequeños, en el mismo sitio
Proceso controlado, con muchas normas y políticas	Proceso menos controlado, con pocos principios
Proceso rígido	Proceso flexible con adaptación
El cliente interactúa con el equipo de desarrollo	El cliente es parte del equipo de desarrollo
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código

Tabla 2– Metodologías Ágiles

http://portal.amelica.org/ameli/jatsRepo/24/2414011/html/

En concreto, para este trabajo, se ha optado por la metodología ágil conocida como *Desarrollo iterativo e incremental* (Antonieza Kuz, 2018). Esta metodología consiste en planificar el desarrollo en distintos bloques temporales, llamados iteraciones, de una duración entre dos y cuatro semanas.

Las iteraciones se han ideado para ir obteniendo resultados constantes e ir ampliando el proyecto progresivamente, es decir, el objetivo final de cada iteración es tener un producto operativo y conforme van pasando las mismas ir dando forma y mejorando el mismo hasta tener el producto final. Lo idóneo para que esta forma de trabajar funcione es fragmentar el proyecto en varias secciones y a su vez cada una de estas en varias tareas pequeñas, de esta forma el trabajo se encuentra organizado, se van realizando y completando tareas cada pocos días, por lo que productividad sube. Además, al tener un prototipo al final de cada iteración se puede hacer una evaluación de este por si hay que realizar correcciones y/o mejoras de las funcionalidades ya implementas o del contenido ya desarrollado

4.1 Herramientas de gestión

En concreto, para la organización de tareas y su evolución se ha optado por el método Kanban junto a la herramienta Trello, la cual nos permite clasificar las tareas mediante el uso de tarjetas en varias tablas en función de su estado actual. Esta clasificación, para que se vea de forma visual, se suele mostrar en forma de tablero, como se ve en la Figura 14.

Figura 14– Tablero clásico kanban

Fuente: https://www.javiergarzas.com/2011/11/kanban.html

A cada tarea se le asigna una duración estimada y una prioridad. Durante el desarrollo de la iteración se va midiendo el tiempo real que cuesta cada tarea y se compara con lo estimado inicialmente, para este control se utilizará la herramienta Clockify. Si se produce un desvío,

tanto al alza como a la baja, se deben recalibrar los tiempos del resto de tareas para llegar a la fecha de fin de iteración con todas las tareas planificadas terminadas. En caso de que esto no sea posible y alguna tarea se retrase, esta pasará a la siguiente iteración con una prioridad superior, ya que es una tarea atrasada. En caso extremo de que una tarea sufra un retraso muy grande o su realización se vea comprometida, se debe realizar una reevaluación del proyecto y considerar la opción de descartar la tarea. Es importante remarcar que el objetivo de este sistema de trabajo se basa, como ya se ha mencionado, en ir creado un prototipo funcional por iteración.

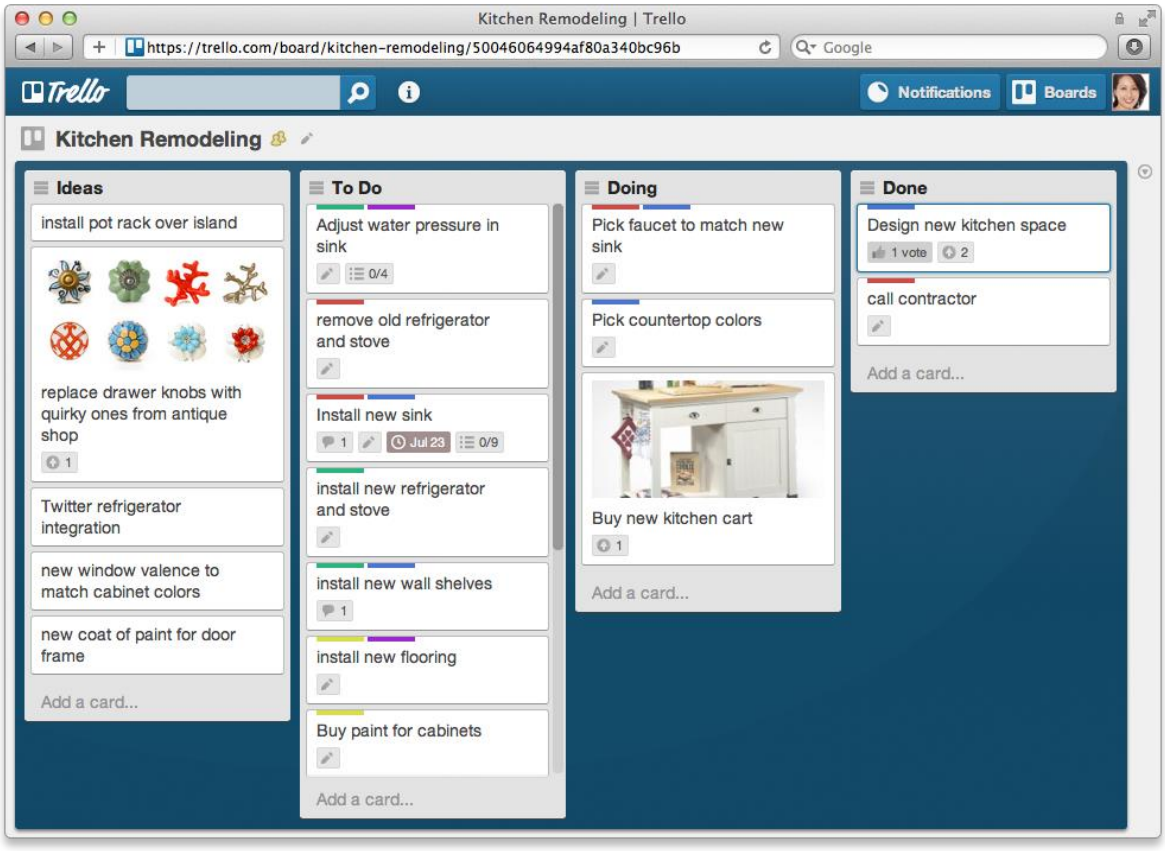


Figura 15– Trello Example Board

Fuente: https://blog.trello.com/engineering-teams-sample-trello-boards

4.2 Control de Versiones

Para realizar el control de versiones, como es normal, se va a utilizar GitHub en conjunto con GitKraken, un programa que nos facilita todas las funcionalidades de GitHub mediante una interfaz visual. El uso de estas herramientas nos va a permitir trabajar con mayor facilidad, pues garantiza acceso al trabajo de forma inmediata y, sobre todo, de forma segura, ya que se realizarán copias de seguridad de forma extremadamente simple a medida que se avance en el proyecto.

El uso de esta herramienta es obligatorio, ya no solo por todas las ventajas que aporta al desarrollo sino porque todas las empresas hoy en día usan esta herramienta, o una muy similar, para almacenar y proteger sus proyectos.

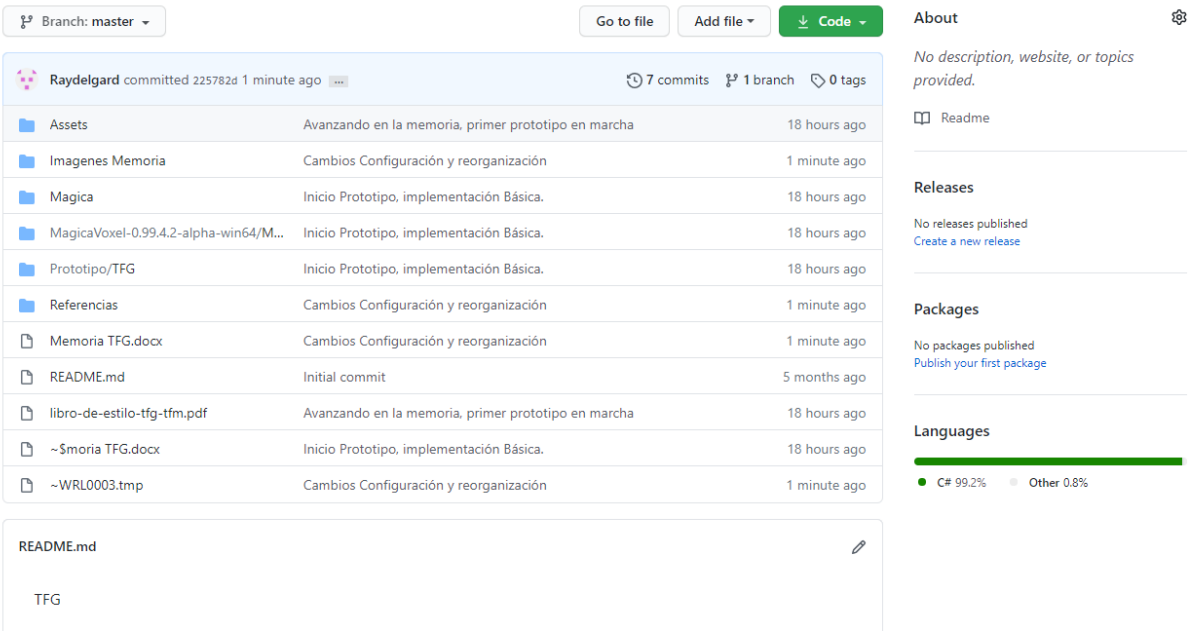


Figura 16– Trello Example Board

Fuente: Elaboración propia

Como el proyecto es de carácter individual no será necesario utilizar muchas de las herramientas de GitHub, para este proyecto será suficiente el uso de los commits, para crear las copias de seguridad y el uso de ramas para la implementación de las distintas funcionalidades. Una vez que una funcionalidad esté operativa se fusionaría con la rama principal, en la que siempre habrá una versión usable del producto.

5. Análisis del proyecto

En este apartado se va a empezar a tratar los temas de forma más técnica y a profundizar en los aspectos más relevantes del trabajo. Se van a realizar análisis sobre diversos temas muy importantes que deber ser realizados previamente al comienzo del diseño del videojuego, tales como la planificación, la gestión de riesgos, software a utilizar y modelos de negocio.

5.1 Planificación Inicial

El objetivo es realizar el mejor trabajo posible dentro de las 300 horas posibles, pare ello, como ya se vio en el apartado de metodología, se va a trabajar siguiendo un modelo iterativo. Para este proyecto se dispone de unas siete semanas. Con esto en mente se ha fraccionado el proyecto en 8 iteraciones diferentes de la siguiente forma. El seguimiento de cada iteración se realizará mediante un tablero de Trello diferente para cada una. Esta división temporal se ha establecido con varios objetivos en mente:

1. Fraccionar el trabajo en pequeñas tareas que permitan avanzar con rapidez.
2. Para este proyecto se ha estimado que siete días es el tiempo mínimo necesario para poder cumplir con los objeticos de cada iteración.
3. Al ser cortas las iteraciones es más fácil realizar cambios si estos son necesarios porque la evaluación del producto es constante.
4. El tiempo disponible para realizar el trabajo no es óptimo así que realizar todas las iteraciones posibles ayudará a tener un mejor producto final.

La planificación fue modificada posteriormente debido a multitud de factores, en las conclusiones del proyecto se detallan, así como la replanificación que se realizó.

5.2 Estudio de viabilidad

En este apartado se va a realizar un estudio completo sobre la viabilidad del proyecto en el contexto actual. Este tipo de análisis es absolutamente crucial hoy en día, cualquier proyecto desde su concepción debe superar varias etapas antes de entrar en producción y una de ellas es el tema que nos atañe en este punto, puesto que un proyecto que se considera inviable será cancelado inmediatamente y nunca verá la luz.

Cualquier estudio de viabilidad debe contar con una serie de puntos para considerarlo eficaz, estos son:

5.2.1 Gestión de riesgos

En todo proyecto es necesario tener en cuenta la muy alta probabilidad de que surjan problemas durante el desarrollo que alteren los plazos o impidan realizar varias tareas. Para minimizar el impacto de estos sucesos, es necesario diseñar con anterioridad el cómo se les hará frente. Este proceso se realiza en 4 pasos (Sommerville, 2005):

1. **Identificación de riesgos**: Determinar todos los potenciales riesgos
2. **Análisis de riesgos:** Clasificar por peligrosidad y probabilidad de que ocurran.
3. **Planificación de los riesgos:** Establecer los planes de actuación
4. **Monitorización de riesgos:** Definir identificadores potenciales

Este sistema de gestión riesgos se encuentra representado en la Figura 20.

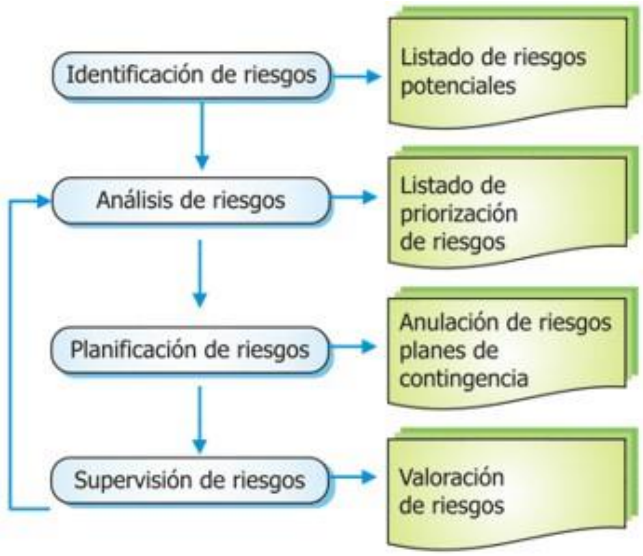


Figura 17- Gestión de Riesgos

Fuente: http://ingesoftwaregestiondelriesgo.blogspot.com/2015/05/blog-post.html

5.2.1.1 Identificación

El primer paso es detectar y listas todos los posibles riesgos, esto es una tarea complicada, por lo que, para facilitar su realización, los riesgos se han clasificado en diversas categorías:

- **Personales**
    - **Enfermedad**: Especialmente en proyectos como este, donde solo hay una persona encargada de todo, este riesgo es muy importante. Más aún dada mi facilidad para enfermar.
    - **Problemas Familiares**: Dado que todavía soy dependiente económicamente de mis familiares, una mala situación familiar puede afectar gravemente al proyecto.
    - **Altos Niveles de estrés**: La carga de trabajo puede llegar a ser muy alta, por lo que hay que intentar mantener unos horarios de trabajo saludables y descansar.
- **Tecnológicos**
    - **Pérdida de información**: La pérdida de datos del proyecto, sobre todo en fases avanzadas del mismo puede ser catastrófico.
    - **Hardware poco potente**: Dado que se va a trabajar con herramientas de muy alto nivel y que necesitan un hardware potente para funcionar de forma correcta es muy importante llevar cuidado al trabajar, especialmente con el motor gráfico.
    - **Problemas con el sistema operativo**: Algo básico y que suele pasar con Windows, hay que realizar mantenimientos periódicos.
    - **Problemas con la conexión a internet**: La conexión a internet es imprescindible para realizar el proyecto, hay que disponer de una alternativa.
    - **Rotura de PC**: Imposibilitaría poder trabajar, obligatorio disponer de un equipo alternativo.

- **Organizativos**
    - **Mala planificación de las iteraciones**: El trabajo de cada iteración debe ser asumible.
    - **Mal fraccionamiento de las tareas**: Las tareas grandes hay que subdividirlas en pequeñas tareas para conseguir avances rápidos

- **Dimensión del proyecto**: Hay que ser realista con el producto real que se puede elaborar en el tiempo disponible, dado que el apartado más relevante es la memoria.
- **No seguir con la metodología elegida**: Kanban nos ayuda a trabajar de forma más eficiente, no hacerlo puede afectar al desarrollo de todas las tareas.
- **No contabilizar las horas de trabajo**: No llevar un registro del trabajo realizado puede llevar a una mala gestión del tiempo disponible.

- **Requerimientos**
  - **Práctica con el nuevo software**: En caso de utilizar un software desconocido es necesario realizar un periodo de práctica para comprender su funcionamiento.
  - **Entrega del proyecto en la fecha indicada**: No entregar el proyecto supone el fracaso absoluto del proyecto.
  - **Cumplimiento de los requisitos**: Realizar el proyecto de forma que cumpla, por lo menos, con los requisitos mínimos establecidos para que pueda ser evaluado.
  - **Funcionalidades básicas**: El proyecto debe incorporar un mínimo de funcionalidades para poder considerarlo un prototipo válido.

- **Estimación**
  - **Infraestimación de las tareas**: Puede afectar a las demás tareas sin una de ellas se retrasa de forma importante.
  - **Sobreestimación las tareas**: Puede afectar al resto de tareas si se dedica demasiado tiempo a una tarea, o demasiado poco.
  - **No tener un producto al final de iteración**: El no cumplir con los objetivos de cada iteración puede alterar severamente el resto de la planificación.
- **Herramientas**
  - **Incompatibilidad entre las herramientas de desarrollo elegidas**: Puedo suponer un retraso muy grave en función del grado de incompatibilidad.
  - **Fallos en el software durante el desarrollo**: Puede ocasionar perdidas de información y del trabajo realizado.
  - **Finalización de las licencias**: Impediría trabajar con la herramienta, afectando al ritmo de trabajo si no se soluciona rápidamente.

5.2.1.2 Análisis

Es este apartado se va a clasificar los riesgos es función de su gravedad y la probabilidad de que sucedan.

Tipo de Riesgo	**Posible Riesgo**	**Probabilidad**	**Gravedad**
Requerimientos	Cumplimiento de los requisitos	Baja	Catastrófico
Requerimientos	Entrega del proyecto en Fecha	Baja	Catastrófico
Requerimientos	Funcionalidades básicas	Baja	Catastrófico
Requerimientos	Aprendizaje nuevo software	Baja	Catastrófico
Tecnológicos	Perdida Información	Baja	Serio
Tecnológicos	Rotura PC	Baja	Serio
Organizativos	Iteración mal planificada	Moderada	Serio
Organizativos	Tareas mal fraccionadas	Moderada	Serio
Organizativos	Cálculo erróneo de la dimensión del proyecto	Moderada	Serio

Estimación	No cumplir los objetivos de las iteraciones	Moderada	Serio
Estimación	Infraestimar tareas	Moderada	Tolerable
Estimación	Sobreestimar tareas	Moderada	Tolerable
Organizativos	No seguir la metodología	Baja	Tolerable
Tecnológicos	Hardware poco potente	Baja	Tolerable
Tecnológicos	Fallo Internet	Bajo	Insignificante
Tecnológicos	Fallo sistema operativo	Bajo	Insignificante
Organizativos	No contabilizar las horas de trabajo.	Baja	Insignificante
Herramientas	Fallo de software	Moderada	Insignificante
Herramientas	Fin de licencias	Baja	Insignificante
Herramientas	Incompatibilidad entre herramientas	Baja	Insignificante

Tabla 3– Análisis de riesgos

Fuente: Realización propia

5.2.1.3 Planificación

Teniendo en cuenta los riesgos anteriores, se van a diseñar estrategias para poder afrontarlos si llegado el momento es necesario hacer frente a alguno de ellos, lo cual es altamente probable. Para hacer frente a aquellos que supongan un riesgo tolerable o superior se han diseñado tres posibles escenarios:

- **Prevención**: Medidas destinadas a evitar que el riesgo suceda.
- **Minimización**: El objetivo de estas medidas es reducir el impacto si el riesgo llega a suceder.
- **Plan de contingencia**: Si el riesgo sucede y su plan de minimización resulta insuficiente, debido a la gravedad, se aplicará este plan.

5.2.1.3.1 Riesgos personales

1. **Problemas familiares**
    a. **Prevención**: Este riesgo escapa a la gestión personal, su prevención no es posible.
    b. **Minimización**: Buscar horarios alternativos de trabajo y reorganizar las tareas
    c. **Plan de contingencia**: Descartar tareas no críticas y enfocar el tiempo disponible en desarrollar un prototipo mínimo.

2. **Enfermedad**
    a. **Prevención**: Tener una alimentación adecuada, dormir un mínimo de 6 horas diarias y hacer algo de deporte.
    b. **Minimización**: Reducir la carga de trabajo durante unos días y guardar reposos el tiempo suficiente, asumiendo enfermedad leve.
    c. **Plan de contingenci**a: En caso de enfermedad grave, suspender el trabajo y seguir las indicaciones médicas temporalmente, en caso de prolongarse en el tiempo la situación, cancelar el trabajo y entregar en otra convocatoria.

3. **Estrés elevado**
    a. **Prevención**: Mantener hábitos saludables, realizar actividades de ocio además de trabajar.
    b. **Minimización**: Reducir la carga de trabajo temporalmente.

c. **Plan de contingencia**: En caso extremo de que el estrés comience a afectar a la salud o causar efectos secundarios, reducir drásticamente la carga de trabajo o suspenderla totalmente hasta que mejore la situación.

5.2.1.3.2 Riesgos tecnológicos

1. **Perdida de información**
    a. **Prevención**: Utilizar las herramientas de control de versiones con regularidad y guardar copias de seguridad extra.
    b. **Minimización**: Volver a la última versión disponible, perdiendo así la menor cantidad de información posible.
    c. **Plan de contingencia**: Este riesgo no requiere de un plan mayor debido a su naturaleza.
2. **Rotura pc**
    a. **Prevención**: Realizar mantenimiento regularmente al hardware del pc, en caso de detectar problemas, realizar copia de seguridad de forma inmediata. Tener un segundo ordenador con el software necesario instalado.
    b. **Minimización**: Trabajar con el pc secundario hasta reparación del principal, si esta es posible.
    c. **Plan de contingencia**: Aunque es complicado, si todos los ordenadores disponibles fallasen, pedir prestado uno o trabajar en un lugar donde los haya de acceso libre, como una biblioteca.
3. **Hardware poco potente**
    a. **Prevención**: Trabajar optimizando lo máximo posible para obtener la mayor fluidez posible a la hora de desarrollar.
    b. **Minimización**: Reducir la calidad del contenido a la hora de desarrollar, utilizar placeholders.
    c. **Plan de contingencia**: En caso extremo de que el hardware disponible fuera insuficiente para siquiera hacer funcionar las herramientas de desarrollo, adquirir mejor equipo si es posible o pedirlo prestado, si esto no fuera posible habría que reevaluar los objetivos del proyecto.

5.2.1.3.3 Riesgos Organizativos

1. **Cálculo erróneo de la dimensión del proyecto**
    a. **Prevención**: Tanto si es por exceso como por déficit, reescalar el trabajo con el asesoramiento del tutor para cumplir los objetivos.
    b. **Minimización**: Eliminar o añadir/mejorar el contenido existente.
    c. **Plan de contingencia**: Si la dimensión del proyecto supera por mucho las capacidades reales de desarrollo se deberán eliminar todas las funcionalidades no críticas para poder tener un prototipo mínimo terminado para la fecha de entrega. Por otro lado, si el proyecto no alcanza los mínimos establecidos, se deberán añadir funcionalidades o profundizar más en los aspectos que sean necesarios de la memoria.
2. **Iteración mal planificada**
    a. **Prevención**: Establecer unos objetivos muy claros, fragmentar bien las tareas y diseñar cada iteración con pequeños objetivos realizables en 7 días.
    b. **Minimización**: Priorizar las tareas críticas y posponer el resto.
    c. **Plan de contingencia**: Priorizar las tareas críticas y eliminar aquellas que no sean necesarias.
3. **Tareas mal fraccionadas**
    a. **Prevención**: Definir previamente a comenzar a desarrollar nada todas las tareas que se deben realizar e investigar y documentarse en profundidad.
    b. **Minimización**: Subdividir las tareas en las que se haya detectado el fallo y reajustar la planificación.
    c. **Plan de contingencia**: Los planes anteriores deberían ser suficientes para tratar cualquier incidencia.
4. **No seguir la metodología**
    a. **Prevención**: Dedicar un rato cada día a actualizar el trello y cuantificar las horas de trabajo correctamente, usar las aplicaciones móviles para mayor comodidad.
    b. **Minimización**: Si se descuida el control del trabajo, parar un día y dedicarlos actualizar y revisar las tareas que se estén realizando o se hayan completado recientemente.

c. **Plan de contingencia**: Los planes anteriores deberían ser suficientes para tratar cualquier incidencia.

5.2.1.3.4 Riesgos Requerimientos

1. **Incumplimiento de requisitos**
    a. **Prevención**: Establecer unos objetivos muy claros, fragmentar bien las tareas y diseñar cada iteración con pequeños objetivos realizables en 7 días.
    b. **Minimización**: Priorizar las tareas críticas y posponer el resto.
    c. **Plan de contingencia**: Priorizar las tareas críticas y eliminar aquellas que no sean necesarias.
2. **Entrega del proyecto en fecha**
    a. **Prevención**: Revisar los progresos.
    b. **Minimización**: Priorizar las tareas críticas y posponer el resto.
    c. **Plan de contingencia**: Priorizar las tareas críticas y eliminar aquellas que no sean necesarias
3. **Funcionalidades básicas**
    a. **Prevención**: Priorizar el desarrollo de estas tareas.
    b. **Minimización**: Posponer tareas menos relevantes.
    c. **Plan de contingencia**: Priorizar las tareas críticas y eliminar aquellas que no sean necesarias o no de tiempo.
4. **Aprendizaje del nuevo software**
    a. **Prevención**: Documentarse bien y obtener recursos de aprendizaje.
    b. **Minimización**: Priorizar las tareas críticas y posponer el resto.
    c. **Plan de contingencia**: Priorizar las tareas críticas y eliminar aquellas que no sean necesarias.

5.2.1.3.5 Riesgos Software

En el caso de este tipo de riesgos no es necesario realizar planes de actuación debido a que su impacto en el proyecto es mínimo, siguiendo los planes de prevención de los riesgos anteriores estos quedan cubiertos. Es más, solo con la realización de un control de versiones correcto y realizar mantenimiento de los equipos de trabajo es más que suficiente para evitar que estos problemas sucedan o en caso de que sucedan no tengan ningún impacto.

5.2.1.4 Monitorización de Riesgos

La monitorización de los riesgos es un factor muy importante para poder prevenir los mismos, para ello es necesario establecer una serie de identificadores clave que nos señalen la posibilidad de que un riesgo suceda. De esta forma se atajarán los riesgos de una forma más segura y sin tener que llegar a los planes de contingencia. (Referenciar Gráfica)

Tipo de Riesgo	**Identificadores**
Personal	• Bajo rendimiento • Problemas de salud • Exceso de trabajo
Tecnológico	• Fallos en el hardware
Herramientas	• Incompatibilidad entre las herramientas de desarrollo • Lentitud a la hora de trabajar con las herramientas
Organizativo	• Fallos a la hora de cumplir con los objetivos de las iteraciones • Incumplimiento de la planificación • Pocas reuniones con el tutor
Requerimientos	• Funcionalidades básicas incompletas • Mala calidad del software • No cumplir con las exigencias del tutor

Tabla 4– Monitorización de riesgos

Fuente: Realización propia

5.3 Estimación de Costes

La estimación de costes consiste en realizar un cálculo aproximado de lo que costará el desarrollo. Para ello se utilizarán técnicas como el pricing to win o la ley de Parkinson.

Para esta estimación se ha establecido un tiempo de desarrollo de 300 horas, el tiempo asignado al trabajo.

5.3.1 Componentes de Costes

En este apartado se detallan todos los gastos que se deben afrontar durante el desarrollo del proyecto. Con el objetivo de intentar ser lo más realistas posibles se dividirán en gastos en fijos, si estos se pagan mensualmente y puntuales si se pagan cada cierto tiempo o una única vez, posteriormente se desglosarán los impuestos.

Tabla de gastos fijos:

GASTO	COSTE
Salario	1200€/mes
Licencias Adobe	60€/mes

Tabla 5– Gastos fijos

Fuente: Realización propia

Gasto total fijo: 1200+60= 1260€ mensuales, incluyendo el salario.

Gasto fijo sin salario: 60€

Descripción gastos fijos:

1. El sueldo de mil doscientos euros equivale a un sueldo de 7,5€ la hora
2. Las licencias de adobe incluyen todos los softwares que se va a utilizar para otras tareas relacionadas. Especificar software

Tabla de gastos puntuales:

GASTO	COSTE
Gastos de publicación	125€
Inversión en hardware	700€

Tabla 6– Gastos puntuales

Fuente: Realización propio

Gastos puntuales totales: 825€.

Descripción gastos puntuales:

1. Los gastos de publicación incluyen las plataformas IOS y Android
2. La inversión en hardware incluye el pc adquirido para poder desarrollar el proyecto

Desgloses impuestos:

Impuesto de Sociedades: Entidades de nueva creación (excepto las que deban tributar a un tipo inferior), que realicen actividades económicas, en el primer período impositivo en que la Base imponible resulte positiva y en el siguiente deben pagar un 15%.

Este impuesto se calcula mediante los beneficios obtenidos, dado que el primer año la actividad económica **no irá acompañada de beneficio económico**, la empresa queda exenta de este impuesto.

Impuesto de Activdades Económicos: Impuesto obligatorio sobre la actividad económica en nuestro caso, como el importe neto de la cifra de negocios en ningún caso será superior al millón de euros, el proyecto queda exento de este impuesto.

IVA: Impuesto obligatorio que en este tipo de productos es actualmente un 21%.

Salarios: La siguiente tabla muestra el desglose de los impuestos que se deberán pagar por trabajador

Sueldo bruto anual	12.000 €
Cuota a pagar en el IRPF	0,0 €
Cuotas a la Seguridad Social	802,3 €
Sueldo neto anual	11.197,7 €
Tipo de retención en el IRPF	0,00

Tabla 7– Puntos fuertes y débiles de los juegos analizados

Fuente: Realización propia

Ley de Parkinson

La estimación de Parkinson (Gdp Master) consiste en tener en cuenta simplemente el tiempo que llevará el desarrollo del proyecto. De esta manera se realiza una estimación muy ajustada y es común que no se terminen los proyectos. Se utiliza esta estimación como guía, ya que esta estimación supone que el proyecto se va a desarrollar de forma ideal y sin problemas que puedan ralentizar el desarrollo, lo cual es extremadamente inusual.
Para realizar los cálculos se asume que el proyecto se realizará en dos meses de forma ininterrumpida.

Se dispone de 8 semanas de las cuales no se contarán los fines de semana, por tanto, en total se cuenta con de 8*5=40 días, en los que si se trabaja 8 horas al día resulta un total de 40*8= 320 horas.

Estableciendo un sueldo de 7.5 € por hora trabajada lo que nos deja con un total de 320*7.5= 2420€.

A este precio de personal habrá que sumarle el resto de los gastos que han sido calculados previamente por cada mes: 60 * 2 + 2420 = 2540€.

A esto se le suma el gasto puntual total 825 + 2420 = **3425€ totales,** teniendo en cuenta que hay que pagar el IVA cuando se venda el producto, para cubrir gastos es necesario vender por valor aproximado de **4144€.**

Pricing To Win

La técnica denominada Pricing to Win (BOOST, 2019) consiste en que el coste del proyecto se determina en base a lo que el cliente esté dispuesto a pagar, esto conlleva una serie de ventajas e inconvenientes. La ventaja más destacable es que la empresa desarrolladora es la que consigue el contrato de dicho proyecto; como inconvenientes la probabilidad de que el cliente obtenga el sistema que quiere es realmente pequeña, además, los costes no reflejan de forma real el trabajo que ha requerido el proyecto.

Dada la naturaleza del proyecto, creemos que es imposible realizar un cálculo mínimamente realista de lo que un supuesto cliente estaría dispuesto a pagar para realizar el proyecto. Por lo que vamos a trabajar de forma distinta. En las conclusiones se explicará todo en detalle.

Lo que haremos con Pricing to Win es utilizar una de sus vertientes, Pricing de Penetración, se explica la técnica en profundidad más adelante, para establecer un precio

de producto basado en datos reales.

Para establecer un precio de producto basándonos en datos reales vamos a realizar un breve estudio sobre la principal referencia del nuestro Videojuego, Monument Valley, analizaremos su precio, sus análisis y la retroalimentación de los usuarios.

Se han consultado las tiendas tanto de IOS como de Android:

- 4.49 en IOS
- 4.49€ en Android

El precio vemos que se ha mantenido en ambas tiendas sin variar, a excepción de ofertas puntuales y momentos en los que el juego estuvo gratuito.

Para estudiar la recepción de los juegos consultaremos los análisis de varios medios especializados en videojuegos y su nota media en metacritic.

El videojuego obtuvo en metacritic un 89/100, en las valoraciones de los medios especializados, estas son muy buenas, destacando su originalidad, arte y música. En cuanto a las valoraciones de los usuarios nos encontramos con un 82/100 una nota muy buena y que muestra como la propuesta, pese a tener un precio elevado, tuvo una gran recepción. Tras analizar los comentarios positivos, encontramos diversos de este estilo "**This game was absolutely surprising. I bought it (I think) around an hour after it was released. The game is beautiful, the soundtrack is pretty good, and the level design is incredibly clever. If you're looking for a good puzzle game on your favored iOS device, I whole heartedly recommend Monument Valley...**" Donde se puede observar claramente que el videojuego llamó la atención de los usuarios en los aspectos que este quería.

Modelos de Negocio disponibles

El modelo de negocio se define, de forma simplificada, como la forma que tiene una empresa de ganar dinero con la venta de un producto. Este concepto se aplica en cualquier industria, con diferencias entre ellas obviamente. Dentro del mercado de los videojuegos se pueden resumir los modelos de negocio en cuatro variantes, de las que veremos sus principales ventajas e inconvenientes y en que plataformas predomina cada una.

1. **Modelo Pay-to-Play:** Este modelo de negocio ha sido el tradicional dentro de la industria de los videojuegos. Consiste en hacer pagar al consumidor un precio para poder adquirir una copia del título, ya sea digital o física, para poder disfrutar del mismo.

   La principal ventaja de este modelo es la rapidez con la que se recupera la inversión realizada para desarrollar el título. Si el título vende bien la inversión se recuperará muy rápido y se obtendrán beneficios. Sin embargo, este modelo de negocio está muy ligado a plataformas como las consolas y pc. Esto se debe a que el público de estas está más interesado en los videojuegos, es más exigente y está dispuesto a gastar dinero, en algunos casos grandes cantidades.

2. **Modelo Free-to-play:** Este modelo de negocio, relativamente nuevo, consiste en ofrecer al consumidor el videojuego completo de forma gratuita. La forma en la que estos títulos generan beneficios es muy distinta y variada en función del género del videojuego, pero en general caerá dentro de alguno de estos dos bloques:

   - Compras dentro del título: Se suele ofrecer a los jugadores compras dentro del videojuego para obtener recompensas únicas o para hacer su progresión más rápida.
   - Anuncios: En algunos juegos es necesario visionar algún anuncio cada determinado tiempo para seguir jugando, este tipo de anuncios pueden ser eliminados gastando algo de dinero en el juego.

   Este modelo de negocio funciona especialmente bien en dispositivos móviles, donde los usuarios están muy poco acostumbrados a invertir dinero para poder acceder al juego, pero si dispuestos a hacerlo a la hora de hacer compras dentro de ellos.

Dentro de este modelo existe una variante llamada **Freemium**, que consiste en lo mismo que la anterior, pero con una diferencia en cuanto al contenido del juego. En el modelo free to play todo el contenido del videojuego es accesible de forma gratuita, sin embargo, en el modelo freemium, hay contenido que para acceder a él es obligatorio gastar dinero.

Cabe destacar que el 74% de todo el dinero invertido en aplicaciones tanto en IOS como en Android se gasta en videojuegos y que esta inversión crece año a año.

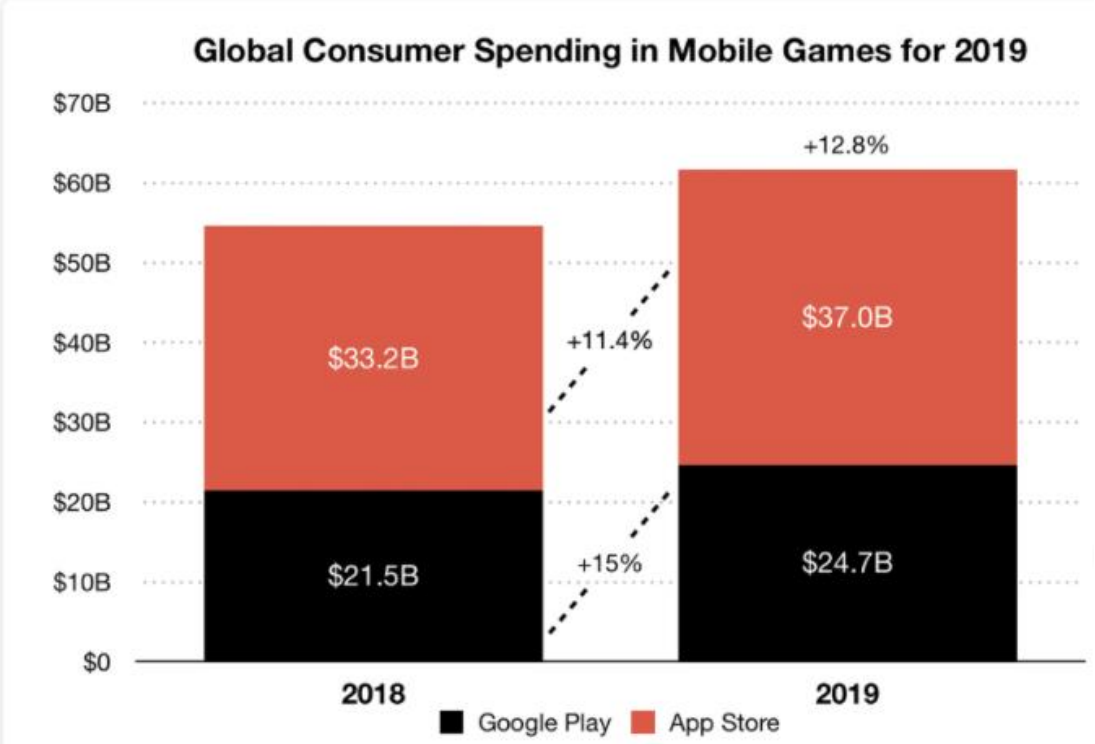


Figura 18- Gasto en videojuegos móviles

Fuente: https://www.unocero.com/noticias/ingresos-y-gastos-juegos-y-apps-app-store-google-play-store-2019/

3. **Modelo Suscripción:** Este modelo de negocio, tradicionalmente, consistía en que el usuario, en primer lugar, debía adquirir el título y posteriormente pagar una, cantidad de dinero cada cierto tiempo, normalmente mensual.

Este modelo de negocio, en lo que a videojuegos se refiere, siempre ha estado muy vinculado a los videojuegos MMO y fuera de este género está prácticamente extinto. Este modelo genera grandes cantidades de ingresos debido a la constante entrada de dinero, pero también es el más difícil de mantener en el tiempo, pues hay que ofrecer contenido de calidad a los usuarios de forma periódica para mantenerlos activos y que de esta forma quieran seguir gastando dinero en el título.

Comparativa y discusión de los valores obtenidos

Analizando estos datos y tras haber aplicado las técnicas del pricing to win y la ley de Parkinson, se ha decidido aplicar una combinación de ambas que se explica a continuación.

En primer lugar, utilizando los datos obtenidos aplicando la ley de Parkinson al proyecto, se van a realizar dos supuestos, uno ideal y uno realista, suponiendo sobrecoste. De esta forma evitamos la principal debilidad de la ley, el asumir que todo va a ir bien.

Dado que el coste Inicial estimado no es muy alto y asumiendo que habrá una desviación al alza, la mejor técnica de negocio que se puede utilizar es El Pricing de Penetración, la cual consiste en un Pricing to Win más agresivo en la que el producto se lanza a un precio inferior al que dicta el mercado para ganar influencia y clientes de la forma más rápida posible. De esta manera la empresa es capaz de hacerse un hueco en un mercado de gran competencia, cabe destacar que esta técnica es arriesgada y sólo válida durante la etapa de crecimiento de la empresa puesto que puede conducir a un círculo vicioso de reducción de precios y a la larga perderá su efectividad.

Con todo lo mencionado anteriormente y habiendo estudiados los posibles modelos de negocio, se concluye que las mejores alternativas son dos:

- Free to play con anuncios: Mayor visibilidad y descargas.
- Pay to Play: Mayor retorno de la inversión

En este caso dadas todas las variables, se considera que la mejora alternativa es optar por el modelo pay to play con un precio reducido, ya que la cantidad de descargas necesarias para producir el número de anuncios que producirían el dinero necesario es demasiado alta, además de que el título, por su género y mecánicas, no da cabida para introducir anuncios.

El precio final se ha decidido que será de 0.99€.

Ahora se realizará una estimación de las ventas necesarias en una **situación Ideal**:

Asumiendo **4144€** como el gasto total **IVA incluido** y teniendo en cuenta el 30% de los ingresos que se llevan las plataformas, obtenido con la ley de Parkinson, y usando Pricing de Penetración como técnica de Negocio y teniendo como objetivo **cubrir gastos**, el número de ventas necesario para cubrir ventas es de **5700** unidades.

Ahora se realizarán una estimación de las ventas necesarias en una situación en la que el coste real es superior al estimado, lo más probable, asumiendo una **desviación al alza de un 10%,** los cálculos son los siguientes.

Asumiendo **4600€** como el gasto total **IVA incluido, teniendo en cuenta el sobrecoste** y teniendo en cuenta el 30% de los ingresos que se llevan las plataformas, obtenido con la ley de Parkinson, la desviación prevista y usando Pricing de Penetración como técnica de Negocio y teniendo como objetivo **cubrir gastos**, el número de ventas necesario para cubrir ventas es de **6300** unidades.

5.4 Análisis DAFO

El análisis DAFO (IPYME), o también conocido como FODA, es un acrónimo de los siguientes términos:

- **D: Debilidades**
- **A: Amenazas**
- **F: Fortalezas**
- **O: Oportunidades**

DAFO es una herramienta de estudio de Marketing muy útil para conocer la situación tanto internas como externas que nos puedan ayudar a la hora de definir las acciones estratégicas necesarias para cumplir con los objetivos fijados. Las características y beneficios principales de este análisis son:

1. **Sencillez y facilidad de aplicación**
2. **Muy concreto y detallado.**
3. **Aporta una visión general del proyecto.**
4. **Ayuda detectar problemas a nivel de organización.**
5. **Anticipa los posibles problemas y crear planes para afrontarlos.**

Este análisis representa los puntos a estudiar en forma de cuadrícula.

Figura 19- Cuadrícula DAFO

Fuente: https://dafo.ipyme.org/Home

5.4.1 Análisis interno

El primer paso del DAFO es realizar un análisis de todos los elementos internos del proyecto que pueden suponer tanto ventajas (fortalezas) como limitaciones (debilidades) a la hora de competir con otros productos.

1. **Fortalezas:**
    a. **Bajo coste de desarrollo:** Aunque las herramientas más potentes son de pago, el abanico de herramientas gratuitas disponibles que nos van a ayudar a desarrollar un producto de calidad profesional sin invertir dinero.
    b. **Género imperecedero:** El género de puzles fue y es uno de los máximos exponentes de la industria, muchos juegos, tanto pequeñas producciones como superproducciones siguen resultando muy exitosos, gustando mucho a los usuarios y cosechando buenas ventas o descargas.
    c. **Pasión por el trabajo que se desarrolla**: Este aspecto ayuda a mantener la motivación muy alta y unos ritmos de trabajo elevados.
    d. **Modelo de negocio**: Gracias al muy reducido coste de desarrollo y que el desarrollo se realizará en paralelo a otras actividades sin presiones económicas se podrá determinar un modelo de negocio cuyo objetivo no sea únicamente obtener beneficios.
    e. **Plataformas objetivo:** Tanto la App Store como la Play Store tienen requisitos no muy exigentes de publicación.

2. **Debilidades**
    a. **Mercado muy competitivo:** Aunque el mercado es enorme, la competencia dentro del mismo es atroz, miles de juegos son publicados cada año y llamar `más la atención es muy complicada.
    b. **Falta de experiencia con las herramientas de desarrollo:** La falta de experiencia puede acarrear problemas durante el mismo y será obligatorio destinar una parte del tiempo disponible a aprender a usarlas.

c. **Tiempo de desarrollo elevado:** Al ser solo una persona trabajando, el tiempo de desarrollo será elevado, ya que crear todos los *assets* gráficos y realizar la implementación llevará mucho tiempo.
d. **Marketing:** Realizar una campaña de marketing que permita dar a conocer el producto, dados los medios de los que se dispone, es imposible realizar una campaña al uso y habrá que buscar vías alternativas.
e. **Falta de experiencia planificando proyectos reales:** La poca experiencia en este ámbito hace prácticamente imposible realizar una planificación acertada en cuanto a tareas a realizar y especialmente en la estimación de la duración de cada tarea.
f. **Exceso de trabajo**: Al ser una persona solo es necesario planificar muy bien el esfuerzo y la exigencia de las tareas a un plazo razonable, para evitar la sobrecarga de horas.

5.4.2 Análisis externo

El siguiente paso es realizar un análisis de todos los elementos externos del proyecto que pueden suponer tanto oportunidades como amenazas a la hora de cumplir los objetivos planeados.

1. **Amenazas:**
    a. **Juegos del mismo estilo:** Aunque no es común que salgan muchos juegos de este estilo, en los últimos años se ha visto un importante incremento de la cantidad de títulos, además, la calidad de estos es en general muy alta y su precio bajo.
    b. **Cambios en las tendencias del mercado:** Aunque un género sea popular, sucede que van apareciendo modas y géneros que durante un tiempo casi monopolizan el mercado. Si esto sucede en épocas cercanas al lanzamiento del juego, puede dañar severamente a la popularidad del juego
    c. **Cambios legislativos para la publicación**: Si se producen cambios de este ámbito, en función de la gravedad de estos, se debería reestudiar el modelo de negocio o las plataformas objetivo.

d. **Mercado muy exigente:** Los consumidores de videojuegos se han vuelto muy exigentes en algunos aspectos, sobre todo en todo lo referente a la visual y al pulido de los títulos.
e. **Compaginar el desarrollo con otras actividades:** Si el resto de las actividades cobraran mucha relevancia o surgieran nuevas necesidades, podría llegar a darse el caso de posponer el desarrollo indefinidamente o suspenderlo de forma definitiva.

2. **Oportunidades:**
    a. **Distribución Mundial Digital:** Los videojuegos para dispositivos móviles tienen una ventaja inigualable por cualquier otra plataforma, si se consigue publicar el producto tanto en la Play Store como en la App Store, algo que no es extremadamente complicad, el juego estará disponible para millones de personas por todo el mundo.
    b. **Mercado en auge:** Los videojuegos es un sector que no para de crecer año tras año, en especial el sector de los videojuegos para dispositivos móviles.
    c. **Aprendizaje:** Para desarrollar el título será necesario aprender mucho de todos los aspectos relativos a desarrollar y publicar un título.
    d. **Situación mundial actual:** La pandemia del COVID-19 ha provocado un aumento exponencial del consumo de videojuegos a nivel mundial debido al confinamiento y al aumento del paso de horas en casa.

5.5 Análisis del software disponible para el desarrollo

En este apartado se analizará el software necesario para realizar el desarrollo y las opciones disponibles.

5.5.1 Análisis de los motores de videojuegos actuales.

Los motores de videojuegos son, actualmente, la principal herramienta de desarrollo de videojuegos, son la base de cualquier desarrollo y la elección de uno es algo extremadamente importante.

Un motor de videojuegos es un software especializado que dispone de una cantidad enorme de herramientas que ayudan a desarrollar un videojuego de forma mucho más rápida y eficiente. De esas herramientas, hay que destacar especialmente las siguientes (Ruela, 2017):

1. **Motor de renderizado**: Ya sea en 2D o 3D es el encargado de mostrar en la pantalla las imágenes, además también realiza cálculos de polígonos, iluminación, efectos visuales entre otras muchísimas cosas.
2. **Motor de Físicas**: Es el encargado de realizar los cálculos de las todas las interacciones físicas entre todos los elementos del videojuego, tales como colisiones o gravedad entre otros muchos elementos.
3. **Editor de Scripts**: Son las herramientas que permiten al desarrollador crear los comportamientos que deben realizar las entidades del videojuego que no están incluidas de forma directa.
4. **Motor de Sonido**: En el encargado de gestionar todos los recursos sonoros, principalmente permite cargar archivos de audio de diferentes formatos y reproducirlos.
5. **Motor de Red:** Permite desarrollar las funcionalidades online necesarias para implementar un modo multijugador.
6. **Gestor de memoria:** Se ocupa de reservar y liberar la memoria que ocupan los assets en tiempo real, en función de si estos se necesitan en el contexto actual o no.

Estos son los elementos principales de los motores de videojuegos, internamente cada uno realizar muchísimas más tareas y cada uno tiene sus particularidades propias, además, muchos de ellos tienen a su disposición, aparte de las herramientas oficiales, herramientas

desarrollados por usuarios que, tanto gratuitas como de pago. A continuación, se analizan los motores de videojuegos más populares.

5.5.1.1 Unity 3D

Unity es un motor muy popular en la comunidad de desarrolladores hoy en día. Es conocido por su versatilidad, su facilidad de uso y por disponer de una versión gratuita muy potente y que apenas se diferencia de la versión de pago.

Aspectos positivos de Unity

Debido a su popularidad, la cantidad de documentación, cursos, tutoriales e información en general es enorme. Toda esta información permite aprender de forma autodidacta, y gratuita.

Permite programar tanto en JavaScript como en C#, ambos lenguajes muy conocidos y similares a otros lenguajes.

La curva de aprendizaje, especialmente para alguien que no parta de cero, no es para nada agresiva, esto se debe principalmente a que dispone de una interfaz que, aunque de primeras puede ser liosa por la cantidad de opciones, está muy bien organizada y es totalmente personalizable.

El sistema de componentes de Unity es una de sus mayores ventajas, permite programar con gran rapidez, crear parámetros editables desde la interfaz gráfica, añadir y eliminar componentes a los objetos entre otras muchas cosas.

Figura 20- Logo Unity

Fuente: https://www.linuxadictos.com/por-fin-llega-el-soporte-oficial-de-unity-a-linux.html

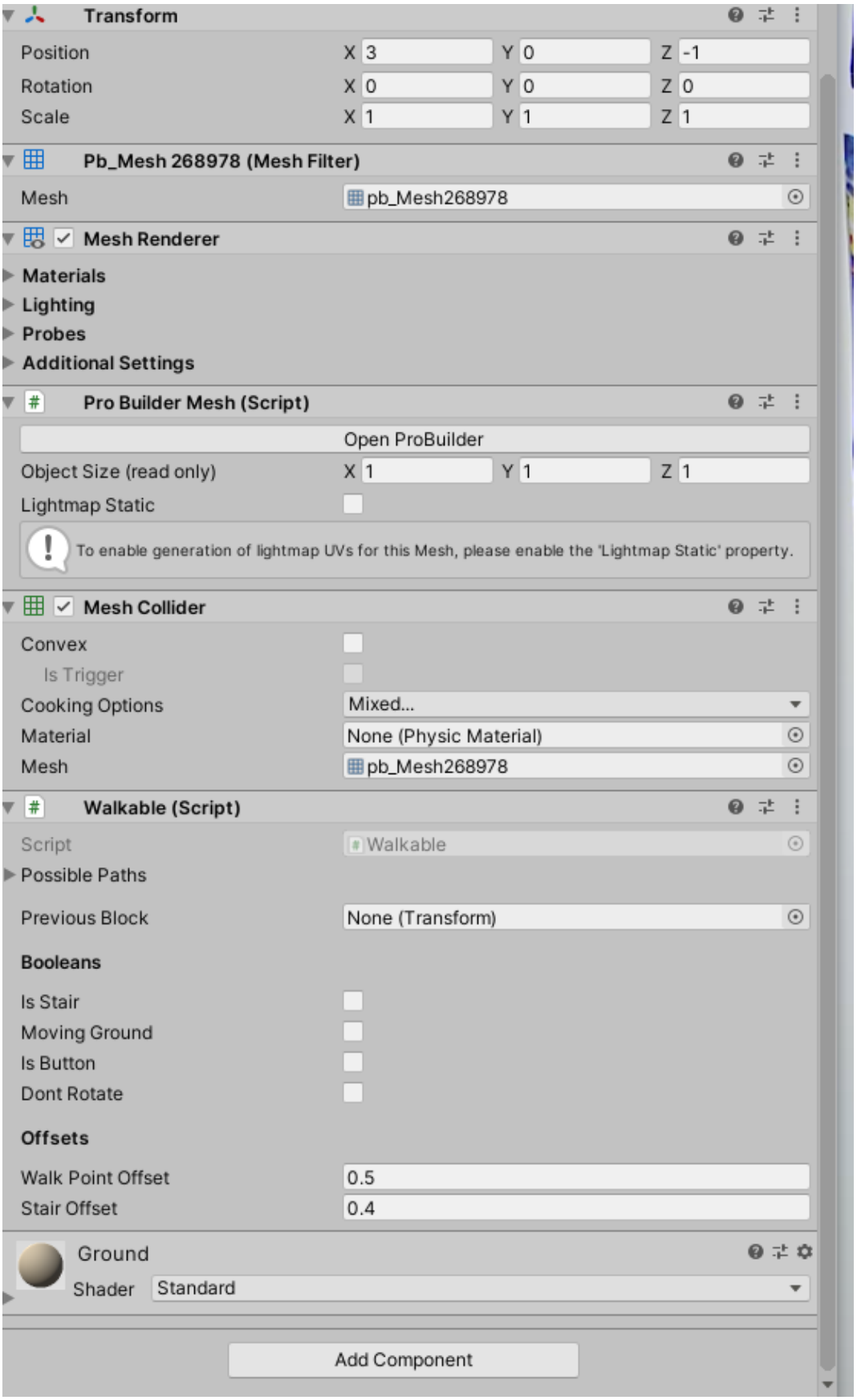


Figura 21-Editor Componentes Unity

Fuente: Elaboración propia

Aspectos negativos de Unity

Debido a la gran potencia del motor, este consume una enorme cantidad de recursos, por lo que es necesario un hardware potente para poder trabajar cómodamente. Además, como la interfaz es muy grande y con varios submenús es prácticamente obligatorio trabajar con monitores de gran tamaño.

Los proyectos, especialmente si se usan assets 3D muy pesados o sonido de alta calidad pesan muchísimo, esto es debido al sistema de ficheros, por lo que, si no se lleva cuidado al importar assets y demás elementos, un juego sencillo puede llegar a pesar varios GB.

Debido a sus grandes virtudes, la cantidad de juego desarrollados en Unity es gigante, el volumen es tal que juegos muy buenos se pierden entre gran cantidad de títulos muy mediocres, esto también es cupla del poco filtro a la hora de publicar juegos, pero sigue siendo un gran impedimento.

Modelo de Negocio y Licencias

Unity dispone de diversos planes, cada uno está pensado para un tipo de usuario, desde planes gratuitos para proyecto muy pequeños sin presupuesto hasta planes para empresas. Las principales diferencias de versiones se limitan al acceso al código y optimización de motor, algo que en general no será demasiado relevante.

Otro de las diferencias importantes es que el plan personal no ofrece acceso a las herramientas de medición de rendimiento. Esto es una lástima porque, aunque C# es un lenguaje eficiente, no está al nivel de C++ y un extra de optimización no estaría de más

	Personal Gratis Comienza a crear con la versión gratuita de Unity Comencemos ¿Eres estudiante? Obtén el plan Student gratuito	Plus 399 $ /año por puesto Más funcionalidad y recursos para potenciar tus proyectos Suscribirse	Pro 1.800 $ /año por puesto Una solución completa para que los profesionales creen, operen y moneticen Suscribirse	Empresa 200 $ /mes por puesto Éxito a gran escala para organizaciones grandes con objetivos ambiciosos Suscribirse Para equipos grandes
ⓘ Requisitos financieros	Elegible si los ingresos o los fondos son inferiores a US$100 mil en los últimos 12 meses	Elegible si los ingresos o los fondos son inferiores a US$200 mil en los últimos 12 meses	Si tus ingresos o fondos han sido superiores a USD 200 mil en los últimos 12 meses, tienes que usar un plan Pro o Enterprise	20 puestos mínimo. Si tus ingresos o fondos han sido superiores a USD 200 mil en los últimos 12 meses, tienes que usar un plan Pro o Enterprise
Crea				
ⓘ Plataforma básica de desarrollo en tiempo real de Unity	✓	✓	✓	✓
ⓘ Herramienta Bolt para scripting visual	✓	✓	✓	✓
ⓘ Personalización de la pantalla de inicio	—	✓	✓	✓
ⓘ Integraciones con las herramientas de colaboración	1, a elección	✓	✓	✓
ⓘ Paquete de assets de arte de alta gama	—	—	✓	✓
ⓘ Capacidad de licencias de Build Server	—	—	+	✓
ⓘ Acceso al código fuente	—	—	+	+
ⓘ Conjuntos de herramientas de soluciones específicas para cada sector de la industria	—	—	—	+
Opera				
ⓘ Cloud Diagnostics Advanced	—	✓	✓	✓
ⓘ Análisis básico	—	✓	✓	✓
ⓘ Analytics: Exporta 50 GB al mes de datos sin procesar	—	—	✓	✓
Monetiza				
ⓘ Unity Ads	✓	✓	✓	✓
ⓘ Complemento de compras dentro de la aplicación	✓	✓	✓	✓
Asistencia y aprendizaje				
ⓘ Soporte técnico	—	—	+	✓
ⓘ Administrador del éxito del cliente	—	—	—	✓
ⓘ Acceso prioritario a Unity Success Advisors	—	—	✓	✓
ⓘ Fila con prioridad para servicio al cliente	—	—	✓	✓
ⓘ Plan de aprendizaje personalizado	—	—	—	✓
ⓘ Sesiones de Learn Live de Enterprise (4)	—	—	—	✓
ⓘ Servicios para el éxito integrados	—	—	+	+
	Gratis Comencemos ¿Eres estudiante? Obtén el plan Student gratuito	399 $ /año por puesto Suscribirse	1.800 $ /año por puesto Suscribirse	200 $ /mes por puesto Suscribirse Para equipos grandes

Figura 22

Fuente: https://unity.com/es

5.5.1.2 Unreal Engine

Unreal es un motor muy popular, utilizado desde el año 2007 y creado por Epic Games. Se utiliza principalmente por grandes empresas, aunque gracias a que ha pasado a ser gratis empieza a ser usado por los usuarios.

Aspectos positivos de Unreal Engine

Unreal es un motor extensamente utilizado y desde que pasó a ser gratuito la cantidad de información disponible y assets ha aumentado mucho, aunque sigue estando muy lejos en este aspecto de Unity. Por otro lado, el motor es de código abierto, por lo que los usuarios pueden desarrollar mejoras por sí mismos.

La mayor virtud de este motor es el sistema de Blueprints, estos permiten programar mediante un sistema de etiquetas visuales sin tener conocimientos previos de programación, lo cual facilita mucho el trabajo.

Utiliza en lenguaje C++, el cual es muy potente y permite extender el motor como quiera el desarrollador, si este sabe cómo hacerlo.

Actualmente, Unreal es a nivel técnico un motor muy potente, pero, además, su nueva versión Unreal 5 ya anunciada, por lo que se ha visto, va a suponer un gran salto a nivel tecnológico.

Figura 23 - Logo Unreal

Fuente: https://commons.wikimedia.org/wiki/File:Unreal_Engine_Logo.svg

Figura 24- Editor Componentes Unreal

Fuente: https://lignux.com/instalar-unreal-engine-4-en-ubun

Aspectos negativos de Unreal Engine

La curva de dificultad para aprender a utilizar Unreal Engine es mucho más elevada, C++ es un lenguaje muy potente pero muy complicado de dominar.

Los blueprints son muy útiles, pero son mucho menos potente y pueden llegar a ser poco intuitivos, especialmente pensado solo para personas que desconozcan programar

La optimización del motor para videojuegos en dispositivos móviles deja bastante que desear, por lo que se recomiendan otras opciones.

El editor de la escena de Unreal, sin ser absoluto malo, carece de algunas opciones muy cómodas a la hora de crear escenarios y manejar todos los objetos de la escena.

Modelo de Negocio y Licencias

Unreal Engine es completamente gratuito y de código abierto, por lo que no hay que pagar nada para usarlo en nuestros proyectos. Lo único que hay tener en cuenta es que si nuestro producto supera los 3000$ de ingresos hay pagar un 5% de los ingresos a la compañía.

5.4.1.3 Cry Engine

Cry Engine es el motor desarrollador por la empresa Crytek, muy utilizado en los últimos años, muy destacado por sus grandes capacidades de simulación de escenarios y efectos en tiempo real.

Aspectos positivos de Cry Engine

Utiliza el lenguaje C# como herramienta principal, además también dispone de un sistema de Scripting llamado Flow Charts, el cual es muy bueno para diseñar eventos y elementos de la inteligencia artificial.

A nivel de potencia visual, es una auténtica bestia, la simulación de eventos visuales, la gestión de la geometría y la calidad general que se puede alcanzar hace de un motor ideal para diseñar grandes escenarios realistas.

Los proyectos, especialmente si se usan assets 3D muy pesados o sonido de alta calidad pesan muchísimo, esto es debido al sistema de ficheros, por lo que, si no se lleva cuidado al importar assets y demás elementos, un juego sencillo puede llegar a pesar varios GB.

Figura 35 - Logo CryEngine

Fuente: https://logodix.com/cryengine

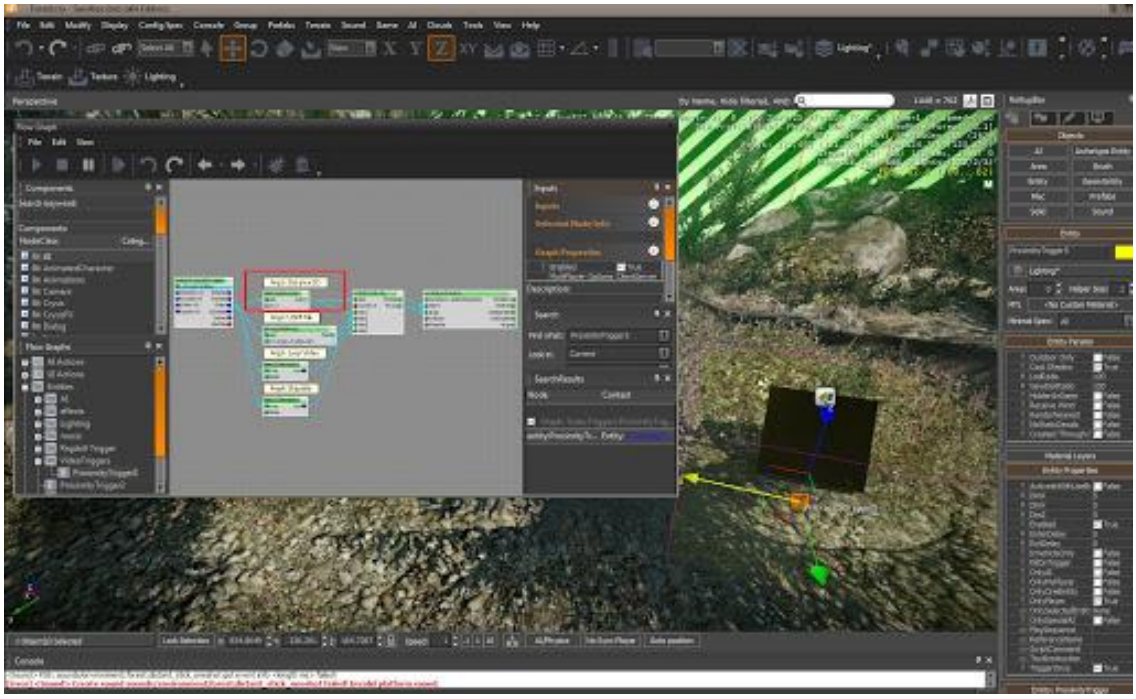


Figura 25-– Editor CryEngine

Fuente: http://www.davevoyles.com/cryengine/

Aspectos negativos de Cry Engine

Cry Engine tiene un problema muy grande, a pesar de ser un motor comercial, apenas se utiliza, y por ello tanto la documentación disponible como la comunidad existente deja mucho que desear y no puede competir con otros motores como Unreal o Unity.

El motor es muy diferente en cuanto a funcionamiento y herramientas con respecto al resto de motores. Esto incrementa su curva de aprendizaje.

Modelo de Negocio y Licencias

Cry Engine ha pasado hace no demasiado a ser completamente gratuito siguiendo el mismo modelo de negocio de Unreal Engine, el desarrollador debe pagar el 5% de los ingresos a partir de los primeros 5000$ dólares.

5.4.1.4 Game Maker Studio

Game Maker Studio es un motor desarrollado por YoYo Games, en concreto por Mark Overmars. Está destinado para usuarios sin conocimientos o con muy pocas nociones de programación y se especializa en videojuegos 2D.

Aspectos positivos de Game Maker Studio

La mayor ventaja de Game Maker Studio es sin lugar a duda su interfaz, que utiliza el sistema *"drag and drop"*, es decir, permite desarrollar videojuegos de forma muy intuitiva organizando iconos en la pantalla. El motor tiene incluidas de serie una buena cantidad de librerías, que además pueden ser extendidas mediante el sistema de generación especial de Bibliotecas.

Game Maker, utiliza un lenguaje de programación propio, GML, inspirado en C, el cual no es muy complicado y permite realizar cosas mucho más avanzadas que las que se permiten de base, algo muy bueno para usuarios experimentados.

La curva de aprendizaje es muy buena, realmente sencillo de aprender y permitiendo realizar títulos en relativamente poco tiempo, pero a su vez con muchas posibilidades de expansión.

Figura 26– Logo Game Maker 2

Fuente: https://www.yoyogames.com/legal/bran

Aspectos negativos de Game Maker

Aunque con conocimientos más avanzados es posible realizar juegos en 3D, el motor está muy especializado en videojuegos 2D.

La licencia gratuita tiene muchísimas limitaciones y no vale la pena, es prácticamente obligatorio comprar la versión de pago.

Solo soporta el lenguaje GML y es débilmente tipado, lo que dificilita la búsqueda de errores y la debugación. Además, no es un lenguaje basado en objetos, sino en eventos, lo que puede causar confusión si se viene de otros motores.

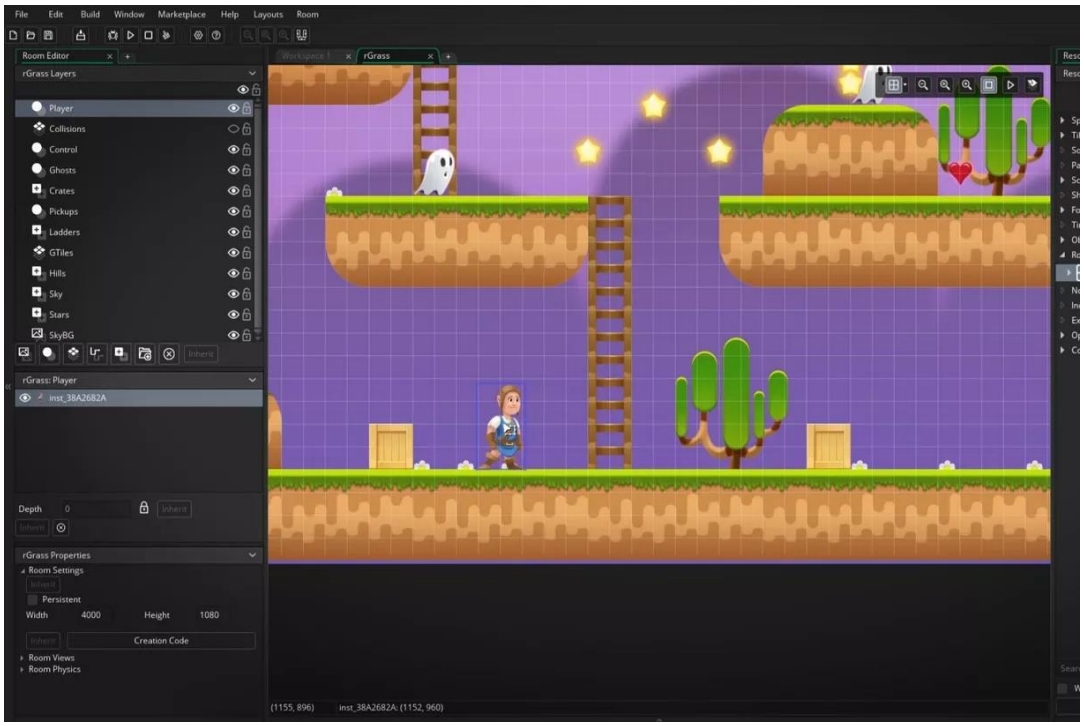


Figura 27– Editor CryEngine

Fuente: http://www.davevoyles.com/cryengine/

Modelo de Negocio y Licencias

El motor dispone, como ya se ha mencionado, de una versión gratuita, pero las limitaciones de esta hacen que no se tenga en cuenta. Es prácticamente obligatorio adquirir su versión de pago, cuyo precio es variable, en función de en cuantas plataformas queramos lanzar el videojuego. Las versiones básicas se encuentran en la figura 38.

Figura 28– Precios Game Maker

Fuente: http://www.davevoyles.com/cryengine/

5.4.1.5 Amazon Lumberyard

Amazon Lumberyard es un motor de muy reciente creación, está basado en el motor CryEngine, con diversas modificaciones como la integración directa con todos los servicios de Amazon, así como con la plataforma Twitch.

Aspectos positivos de Amazon Lumberyard

El motor es totalmente gratuito y ofrece una buena cantidad de herramientas a los desarrolladores, lo cual es sorprende debido a su reciente lanzamiento. Además, permite desarrollar para todas las plataformas existentes, incluida la realidad virtual.

El código es libre, por lo que cualquier persona con conocimientos en C++ puede ampliar y mejorar el motor.

Al estar basado en CryEngine, mantiene muchas de sus características, por lo que al igual que este, Lumberyard es muy potente en cuanto a simulación de eventos visuales, la gestión de la geometría y la calidad general que se puede alcanzar a nivel técnico.

Figura 29– Logo Lumberyard

Fuente: http://www.davevoyles.com/cryengine/

Aspectos negativos de Amazon Lumberyard

El motor fue lanzado hace poco tiempo y apenas hay títulos lanzados con este motor, por lo que la documentación y en especial la comunidad de la que disponen están muy alejadas de todos los motores actuales.

Otro de los grandes problemas es que no soporta videojuegos en 2D, esto es debido a que como ya mencionamos está basado en CryEngine, por lo que está pensado para mundo abiertos y gráficos realistas.

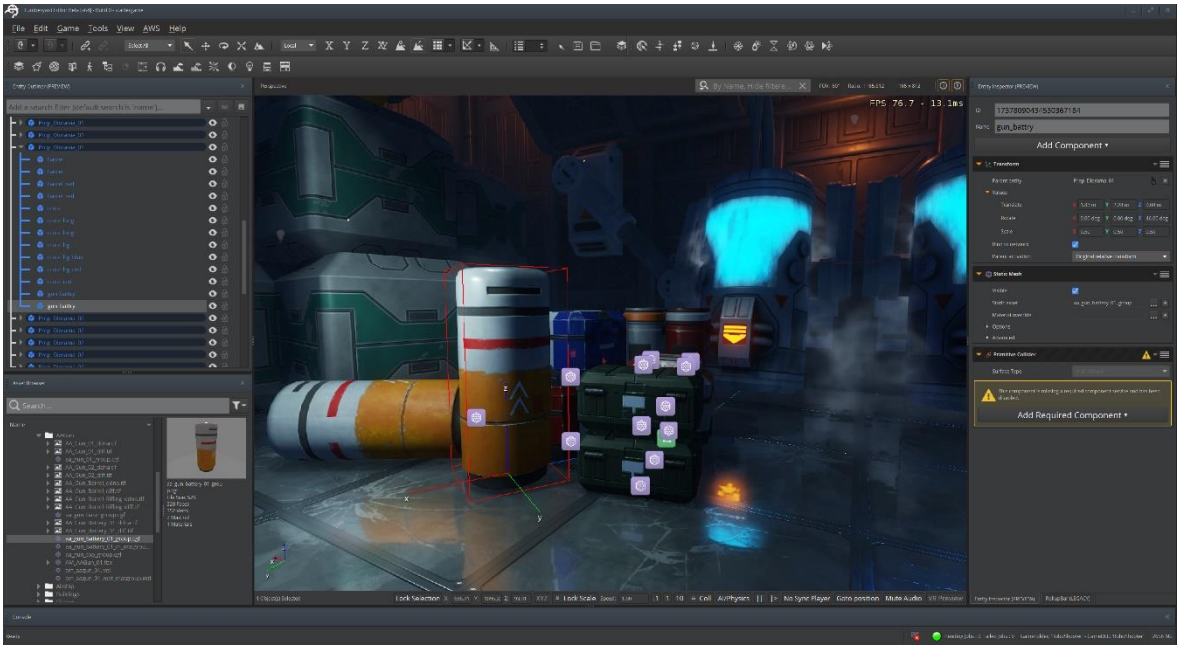


Figura 30– Editor Lumberyard

Fuente: http://www.davevoyles.com/cryengine/

Modelo de Negocio y Licencias

El motor dispone, como ya se ha mencionado, es totalmente gratuito y su código es modificable, siempre siguiendo las restricciones de los términos de uso. Lo único por lo que se debe pagar es por utilizar los diferentes servicios de Amazon Web Services. A diferencia del resto de motores, tampoco es necesario pagar ningún porcentaje de los beneficios a la empresa.

Conclusión

Tras analizar todos los motores gráficos disponibles, valorar los pros y contras de cada uno y teniendo en cuenta las características del juego, el motor gráfico que más se adecua a estas es Unity 3D, por lo que será el motor utilizado para desarrollar el juego.

5.5.2 Otros programas

Además del motor gráfico para desarrollar un juego son necesarios otros muchos programas. Para no extender demasiado este apartado, se van a nombrar el resto de los programas utilizados sin entrar a valorar otras opciones disponibles.

Modelado

Para realizar los modelados que servirán como diseños de referencia de los escenarios, se utilizará la herramienta de modelado MagicaVoxel, la cual es open source y cuyas particularidades son las siguientes:

- Rendimiento increíble y un consumo muy liviano.
- Interfaz muy accesible y sencilla de utilizar.
- Curva de aprendizaje mucho más suave que otros programas de modelado.
- 100% orientada a voxelart, algo idóneo para nuestro proyecto.
- Paletas de colores variadas y totalmente personalizables, hacer algo bonito es relativamente sencillo con poca práctica.
- Motor de render con muchas opciones y posibilidades.

Es sencillo obtener resultados visualmente vistosos en poco tiempo, por ejemplo:

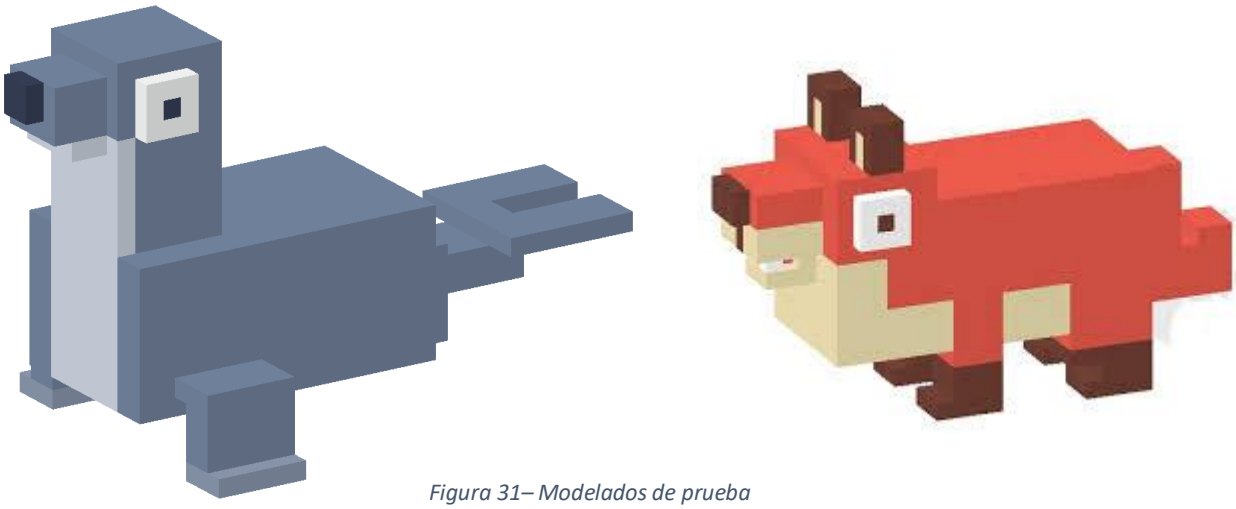


Figura 31– Modelados de prueba

Fuente: Elaboración propia

Tras unas horas de prácticas ya pude realizar un escenario como este, basado en Monument Valley

Figura 32– Escenario realizado para practicar

Fuente: Elaboración propia

El software permite exportar directamente los escenarios para ser importados más tarde directamente en Unity, pero por motivos de optimización de la malla y con el objetivo de facilitar el trabajo a la hora de programar, los escenarios se reconstruyen usando un plugin de Unity.

5.6 Plataformas y publicación

Como ya se ha mencionado varias veces, al ser un videojuego para dispositivos móviles las plataformas objetivo son, obviamente, la app store de IOS y la play store de Android.

Android

El proceso en sencillo, pero consta de múltiples pasos y es largo, a continuación, se explica el proceso, el cual está perfectamente detallado en la página oficial de Googleplay.

El primer paso es crear y configurar una app en nuestro perfil de desarrollador.

- Crear una cuenta de desarrollador y entrar en Play Console, esto requiere un único pago de por vida de 25 dólares.
- Rellenar una serie de apartados en los que debemos indicar el idioma, el título, si la aplicación es una app o un juego y si es de pago o gratuita.
- Debemos proporcionar un correo de contacto disponible para los usuarios.
- Aceptar una serie de contratos y crear.

El siguiente paso es configurar la app, donde debemos proporcionar información detallada de la aplicación y crear la ficha que aparecerá en la play store. Existen muchos parámetros a configurar en relación con los archivos de la aplicación, limitaciones de tamaño, si la aplicación tiene anuncios y una gran cantidad de parámetros, además de otro tipo de restricciones y términos de usuario que hay que aceptar. Para no extender demasiado este apartado, los siguientes pasos se van a resumir en una serie de puntos, estos son:

- Subir al store una versión compilada de la aplicación, el apk.
- Detallar el estado de la versión lanzada, ya que es posible, subir un apk incompleto como borrador para preparar el lanzamiento o para realizar testing.
- Detallar la clasificación de contenido, como el código PEGI, por ejemplo.
- Detallar el contenido de la aplicación.
- Elegir el precio de la aplicación si es de pago

Figura 33– Escenario realizado para practicar

Fuente: Elaboración propia

IOS

La publicación en la app store es bastante más tediosa y larga que en la app store, por lo que al igual que en el apartado anterior de Android, el proceso se va a resumir, mucho, para no extender este apartado.

- Crear una serie de certificados a través de la cuenta de desarrollador.
- Registrar un dispositivo y vincularlo a la cuenta de desarrollador.
- Generar un ID de aplicación, formado por dos partes, una se autogenera y la otra la elige el usuario.
- Crear un perfil de aprovisionamiento, el cual unifica todos los permisos, certificados y dispositivos en un único perfil.

El siguiente paso es configurar iTunnes Connect, esto es imprescindible para publicar la aplicación, a esta característica se accede desde el mismo perfil de desarrollador. En este punto, si la aplicación es de pago debemos solicitar un contrato y aceptar sus términos y condiciones. Además, también se solicitará información bancaria para los ingresos obtenidos con las ventas, cabe destacar que Apple se queda un 30% y un contacto con el responsable de estos términos. Con esto ya se habrá creado una entrada para la aplicación, los siguientes pasos para finalizar la publicación son:

- Crear una ficha de la aplicación con los datos de esta, esta será la información que verá el usuario.
- Proporcionar información sobre los derechos de autor y los datos de contacto del responsable.

- Detallar la clasificación de contenido, como el código PEGI, por ejemplo.
- Proporcionar información sobre la persona responsable de la aplicación ante Apple.
- Una vez finalizados todos los pasos anteriores la aplicación será publicada si supera el proceso de revisión manual de Apple, que actualmente tarda unos dos días en evaluar la misma.

Figura 34– Escenario realizado para practicar

Fuente: Elaboración propia

6.Diseño del proyecto (GDD)

En este apartado se va a hablar de todo lo relacionado con las características del software. Para ello, se va a realizar un GDD, es decir, el documento en que se registran todas las decisiones de diseño del juego.

6.1 Información general del proyecto

El videojuego cuyo nombre se ha decidido que sea **To the Heaven** en un juego de puzles en el que el jugador deberá ir resolviendo una serie de retos en los diferentes niveles, cada nivel tendrá su propio estilo, vistosos visualmente, dando mucha importancia al diseño y al "look and feel" que se detallará más adelante.

6.2 Género

EL videojuego pertenece al género de puzles.

6.3 Publico objetivo

Este videojuego está dirigido a cualquier jugador que disfrute de los videojuegos de puzles, el rango de edad de los jugadores es, dadas las características del título, muy amplio, ya que las habilidades requeridas para jugar son mínimas.

6.4 Resumen del ciclo de juego y estados

El ciclo de juego se compone de las fases que tiene el videojuego y de cómo se desarrollan:

1. El jugador comienza a jugar.
2. El jugador se enfrenta al reto que le propone el nivel en cuestión (Puzle).
3. El jugador llega al final del nivel y pasa al siguiente.
4. Repetir pasos 2 y 3 hasta completar el juego o salir del mismo.

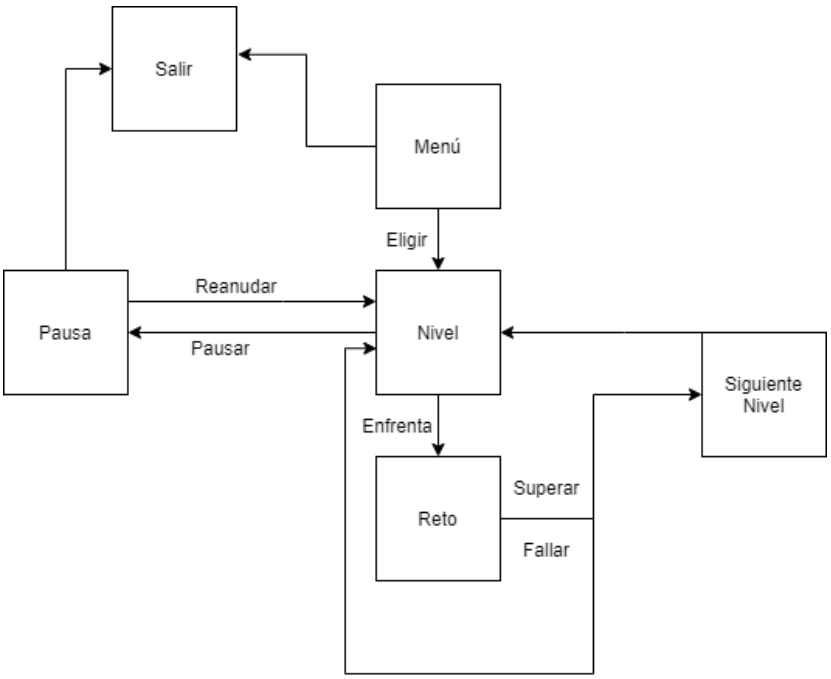


Figura 35– Ciclo del Juego básico

Fuente: Elaboración Propia

En cualquier momento el jugador puede pulsar el botón de pausa y salir del juego. En el apartado de estados del juego se proporciona un diagrama más completo.

6.5 Look and Feel

El look and feel hace referencia al cómo es el aspecto visual del título y si tiene alguna intención en concreto, como transmitir algún sentimiento o narrar algún hecho.

En To the Heaven el objetivo es transmitirle al jugador buenas sensaciones, en especial dos de ellas, calma y relajación. Para ello se han tomado diversas decisiones de diseño que veremos más adelante.

6.6 Alcance del proyecto

El alcance del proyecto se entiende como la cantidad de contenido de la que va a disponer el proyecto:

1. Un único personaje jugable.
2. Tres mundos que agruparan niveles.
3. Diferentes desafíos en forma de puzles a lo largo del juego.
4. Historia muy simple.
5. Unos 15 niveles, aproximadamente.
6. Música y efectos de sonido.

6.7 Estilo Visual

Como se mencionó en el apartado de referencias, el juego se inspira visualmente en obras como Monument Valley, nekorama o Dream Machine. Todos ellos comparten muchos aspectos en lo referente a su estilo.

En primer lugar, lo más importante y lo que hay que destacar por encima de todo, es el uso de los colores. En todas las obras anteriores este es un punto vital, y en To the Heaven el objetivo es el mismo. Para ello, se han estudiado las paletas de colores y lo que transmiten, en el aparatado de niveles, donde se detalla cada nivel, se especifica, además de información sobre el nivel, su paleta de colores específica.

El motivo por el que se ha decidido este estilo es porque el objetivo del juego es ofrecer una experiencia visual atractiva, que llame su atención y sea el principal foco de atención del jugador. Las paletas de color utilizadas son las siguientes.

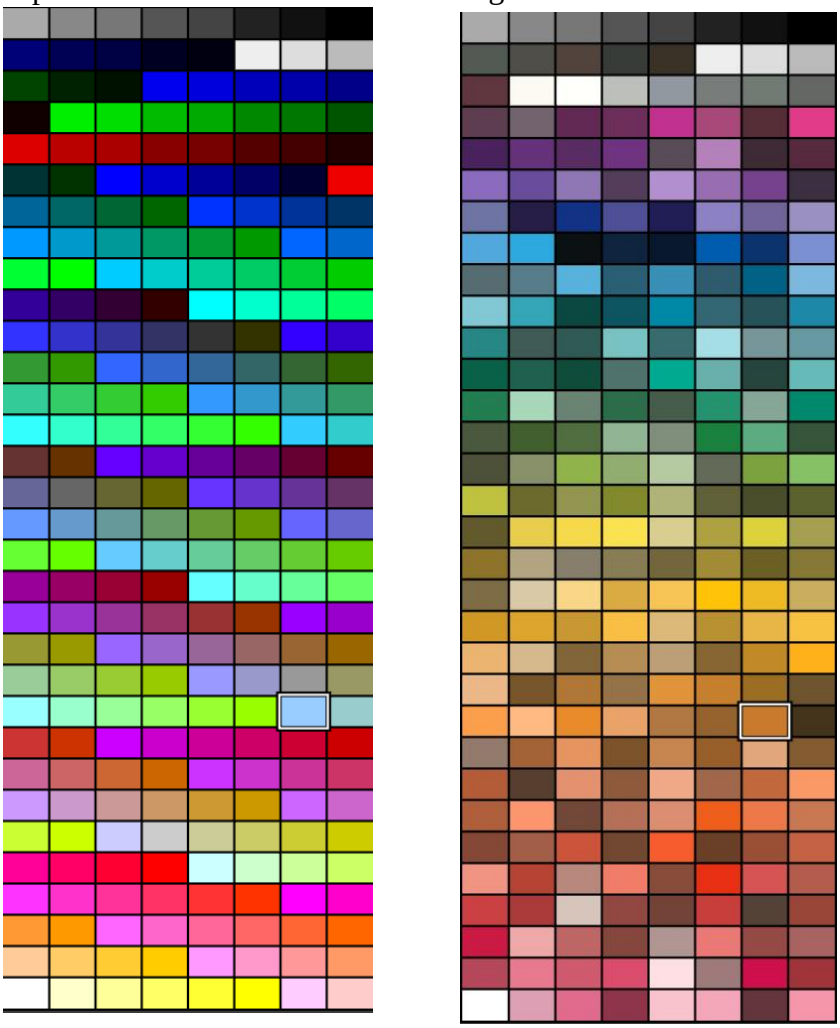


Figura 36- Paletas de colores

Fuente: Elaboración Propia

En concreto, para to the Heaven se han diseñado tres temáticas que agruparan distintos grupos de niveles. Las tres temáticas se muestran con el nivel prototipo.

1. **Infierno**: Ambientación oscura, destacando los tonos rojos, naranjas y negros, un mar de lava rodeará los niveles, habrá enemigos que deberemos evitar. Se han elegido estos colores porque el objetivo es transmitir la sensación de peligro.

Figura 37– Ambientación Infierno

Fuente: Elaboración Propia

2. **Jardines**: Ambientación alegre, destacando los tonos verdes, azules, y grises un mar de agua rodeará los niveles, aquí no habrá enemigos que evitar. Se han elegido estos colores porque el objetivo es transmitir la sensación de tranquilidad.

Figura 38– Ambientación Jardines

Fuente: Elaboración Propia

3. Cielo: Ambientación, destacando los tonos blancos, dorados, y azules, un mar de nubes rodeará los niveles, aquí no habrá enemigos que evitar. Se han elegido estos colores porque el objetivo es transmitir la sensación de esperanza.

Figura 39– Ambientación cielo

Fuente: Elaboración Propia

6.8 Jugabilidad y Mecánicas

En este apartado veremos los aspectos jugables principales del título.

Jugabilidad

La jugabilidad es un concepto muy amplio, pero dentro de ella podemos englobar dos conceptos especialmente muy importantes:

Usabilidad: La usabilidad es la facilidad con la que el juego se puede jugar, esta es resultado de una combinación de elementos como la interfaz, los controles y otros elementos menos relevantes. Respecto a los otros dos, tanto la interfaz del juego como los controles de este, se han reducido al mínimo posibles para que sean extremadamente sencillos para que al jugador le resulten automáticos de aprender y no le distraigan mientras juega.

Experiencia de Juego: La experiencia de juego se refiere a las sensaciones percibidas por el jugador mientras juega, estas pueden ser muy diferentes en función de lo que desee el jugador. En este caso, como ya hemos mencionado, el objetivo del juego es que el jugador juegue relajado, por lo que se han realizado múltiples decisiones de diseño en lo referente a mecánicas, efectos y demás aspectos para ello. Todas estas decisiones se detallan más adelante

Mecánicas

Las mecánicas de un videojuego son todas las interacciones que realiza el jugador para poder interactuar con el mismo. Esto es algo único de cada juego, pero en todos los casos, estas interacciones están controladas por una serie de reglas. En To the Heaven son las siguientes:

Movimiento: El jugador deberá tocar la pantalla para controlar el movimiento del personaje mediante toques, es decir, una vez el jugador toque un punto de la pantalla el personaje se moverá de forma automática siguiendo el camino disponible, en caso de no haber camino disponible o que el camino este solo disponible parcialmente el personaje realizará el movimiento disponible. El movimiento del personaje está restringido a unos caminos preestablecidos.

Mover elementos: Algunos elementos del escenario permitirán interactuar con ellos, estos elementos al ser tocados realizarán una seria de acciones predefinidas, como moverse o girar para desbloquear un camino no disponible. Estos elementos podrán ser activados por el personaje in-game o por el propio jugador.

Pausar el juego: El jugador podrá pausar el juego cuando desee y volver a la partida o al menú principal.

Navegar por los menús. El jugador, si se encuentra fuera del ciclo de juego, significa que está en algún menú, por lo que podrá navegar por ellos.

Enemigos: En algunas zonas del juego habrá una serie de enemigos que se interpondrán en tu camino y que deberás evitar.

Cámara: La cámara es un factor muy importante en los videojuegos, y lo es aún más en To the Heaven. El titulo utilizas las ilusiones ópticas para plantear los puzles, para ellos se ha dispuesto una cámara isométrica que juega con las perspectivas y los renderizados de los objetos para la ceración de los puzles y diversos elementos.

Guardado y Carga: El jugador podrá continuar la partida desde el ultimo nivel donde lo dejo o volver a empezar desde el principio.

6.9 Historia y Personajes

En To the Heaven, la historia narra la historia de Wanda, una niña que emprende un viaje para rescatar a su madre. Para ello deberá emprender un viaje que la llevará por lugares inhóspitos en los que deberá superar diversos desafíos.

La historia comenzará con una pequeña cinemática en la que veremos como la madre de Wanda es secuestrada. Así dará comienzo la aventura y nuestra protagonista es la siguiente.

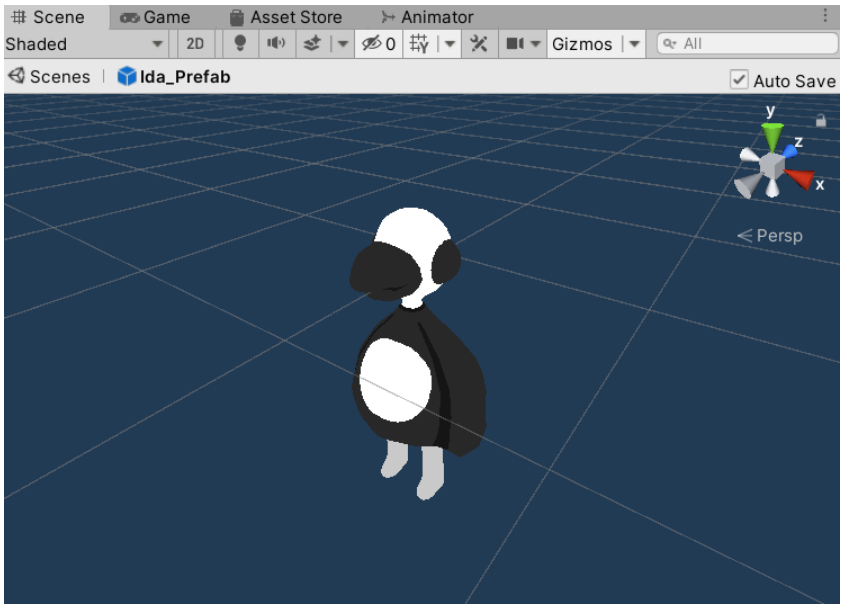


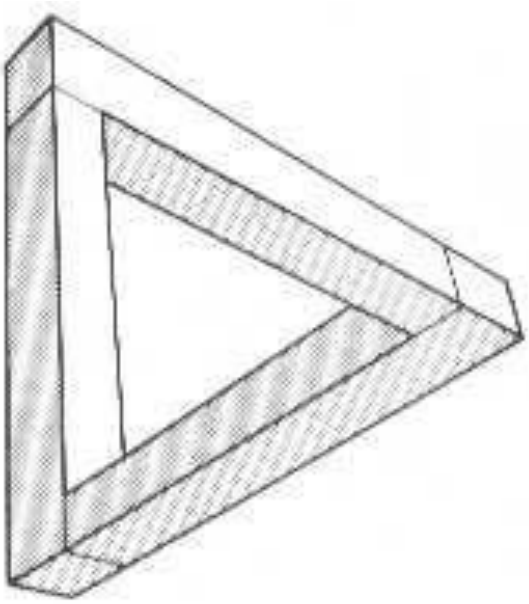
Figura 40– Protagonista

Fuente: Material Cedido

6.10 Ilusiones ópticas

Las ilusiones ópticas que se utilizan en el juego en los puzles están inspiradas en las obras y figuras imposible de Maurits Cornelis Escher (Maurits Cornelis Escher, 2019). To The Heaven combina las ideas de Escher con la utilización de la cámara para generar estar ilusiones.

Por ejemplo, esta es una de las figuras de Escher que aparece en el título.



Cuya aplicación se puede ver ya en el prototipo.

Figura 41-Puzle

Fuente: Elaboración Propio

6.11 Interfaz

La interfaz de To the Heaven será extremadamente sencilla, eliminando todos los elementos que no sean estrictamente necesarios, a continuación, se muestran los mockups de las interfaces del juego, explicando en qué situación se da cada uno de ellos:

Figura 42– Pantalla inicio

Fuente: Elaboración Propia

Al iniciar el juego se mostrará lo que será el nivel 1 de fondo y algún mensaje en caso de que es nivel lo requiera, como, por ejemplo, “pulsa para moverte”

Figura 43– Pantalla pausa

Fuente: Elaboración Propia

Dentro de cualquier nivel el jugador verá como única interfaz el botón de pausa, que al ser pulsado detendrá el juego.

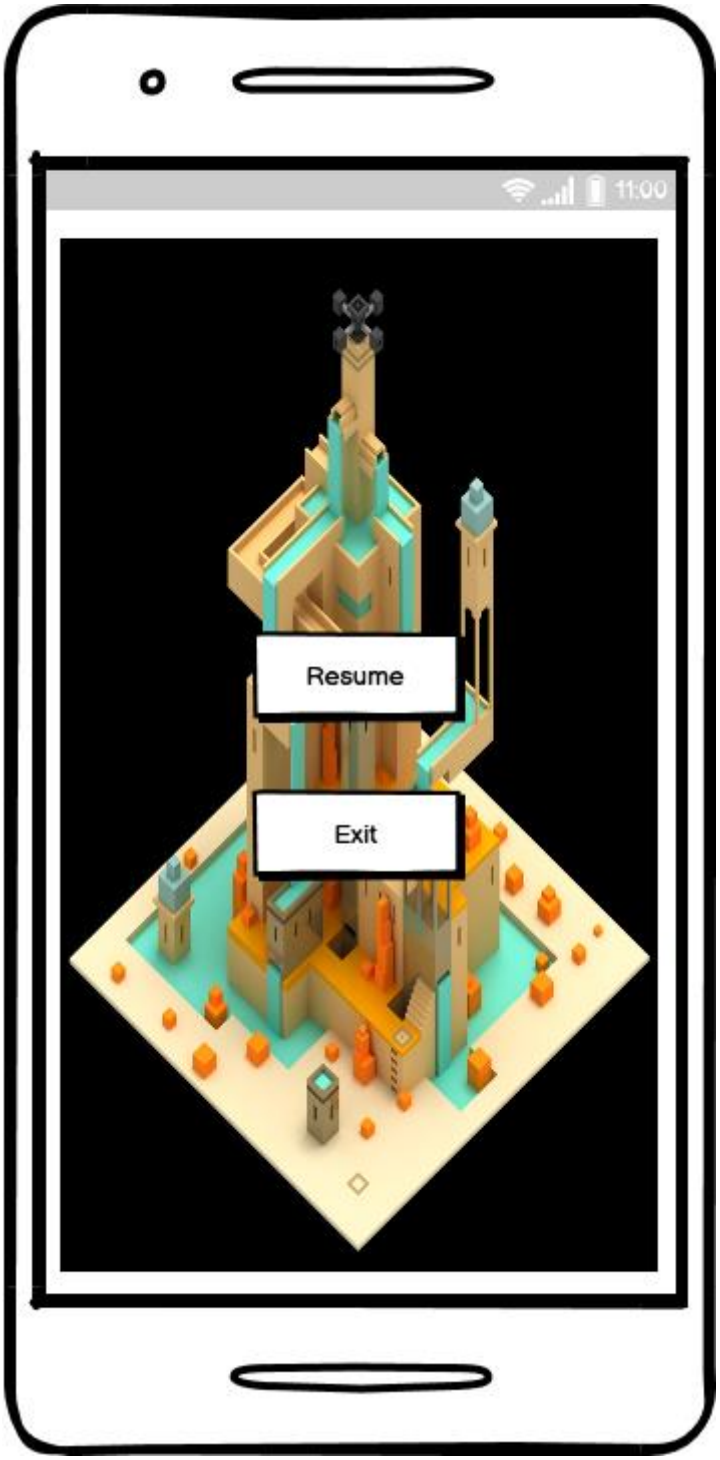


Figura 44– Pantalla pausa

Fuente: Elaboración Propia

Dentro del menú de pausa verá dos botones, continuar y salir. El botón continuar volverá al juego y el botón salir cerrará el juego.

Figura 45– Pantalla Menu

Fuente: Elaboración Propia

En el menú aparecerán los niveles disponibles en los que el jugador podrá elegir el nivel dentro de un selector y comenzar a jugar. También podrá elegir salir de la aplicación pulsando el botón correspondiente.

6.12 Sonido, Música Y Efectos

Este es un aspecto muy importante en cualquier juego, en el caso de To the Heaven, como ya se comentó, el juego se ha pensado para ser una experiencia muy visual y tranquila, por lo que la música deberá acompañar a estas sensaciones. En función del mundo en el que nos encontremos la música cambiará, adaptándose así a las diferencias entre estos.

En cuanto a los efectos de sonido, no habrá, el único sonido será la música ambiente.

Los audios para la música de fondo pertenecen a Patrick de Arteaga.

Figura 46- Nivel Prototipo

Fuente: https://patrickdearteaga.com/es/musica-epica-orquestal-fantasia-medieval/

6.13 Niveles

Como se mencionó en el apartado estilo visual, los niveles estarán agrupados en diferentes mundos con tres temáticas visuales, cada una de ellas corresponde a una fase del juego. En primer lugar, encontramos lo que serán del mundo “Inferno” que hará las veces de tutorial, con niveles más cortos para que el jugador aprenda las mecánicas del juego.

El juego constará de 4 niveles, aparte del nivel prototipo que se mostrará a continuación, uno que hará tutorial y uno perteneciente a cada una de las ambientaciones: infierno, jardines y cielo.

6.13.1 Nivel prototipo

Este nivel, cuyo diseño es una simplificación de un nivel de Monument Valley (Ustwo Games, 2014), diseño creado como banco de pruebas. En él se han desarrollado las mecánicas de movimiento, pathfinding, interacción con los elementos, además de un sinfín de elementos visuales de partículas y reflejos. Más adelante, en el apartado de implementación se explicarán todos estos elementos.

Figura 47- Nivel Prototipo

Fuente: Elaboración Propia

7.Implementación

En este apartado, se va a hablar de como se ha realizado el proyecto a nivel de programación. Se detallan los pasos seguidos, todos los elementos implementados y aquellos que han sido descartados, especificando las razones de su exclusión. Pero, Antes de ello se va a realizar una introducción de algunos elementos de Unity:

7.1 Funcionamiento de Unity

Unity es un motor que integra una inmensa cantidad de herramientas para facilitar el desarrollo, herramientas que en cada versión se van puliendo y mejorando. En este caso, la versión utilizada para el desarrollo de videojuegos ha sido la versión 2019.3.14.

Para no eternizar la explicación de Unity, se ha realizado una clasificación de los elementos más relevantes a la hora de desarrollar y que serán los que más se referencien durante esta sección del documento, estos son:

1. Escenas
2. Assets
3. Scripts
4. GameObjects
5. Componentes
6. Prefabs

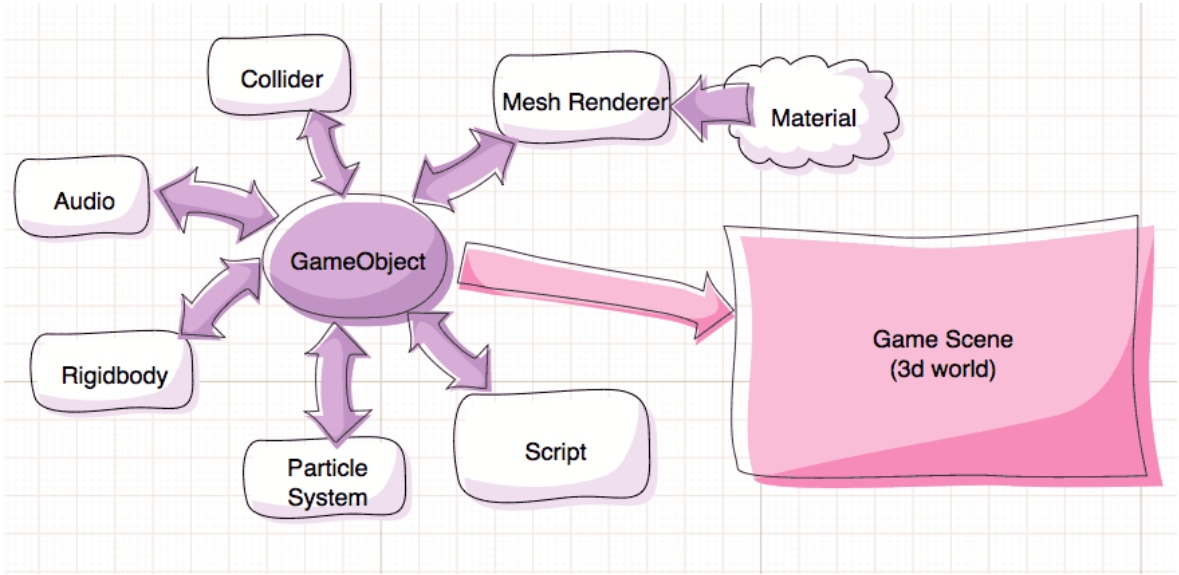


Figura 48– Pantalla Menú

Fuente: https://koenig-media.raywenderlich.com/uploads/2011/08/unity14_diagram.png

7.1.1 Escenas

Las escenas son el elemento primario dentro del motor, todos los demás elementos quedan englobados dentro de ellas. Los proyectos están formados por diversas escenas que se van cargando o descargando.

La funcionalidad de las escenas es hacer de estructura, organizar el resto de los elementos dentro de ellas y guardar la información dentro de ellos. El resto de los elementos mencionados anteriormente y que veremos a continuación se incluyen en la escena.

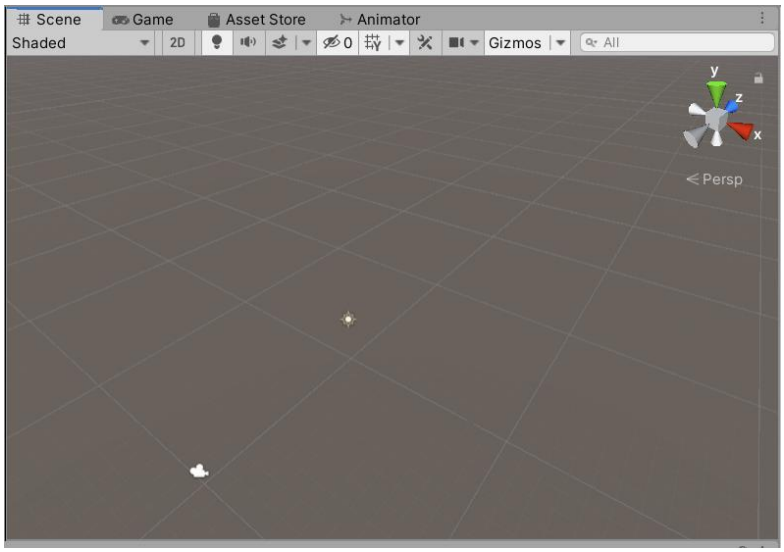


Figura 49-Scene

Fuente: Elaboración Propia

7.1.2 Assets

Los assets son todos y cada uno de los elementos multimedia que aparecen en el juego, como los modelados, imágenes, audios, scripts…Con todos estos elementos es con lo que se crean las piezas del videojuego. Todos estos archivos están organizados en una estructura de carpetas.

Figura 50–Assets

Fuente: Elaboración Propia

Además de estos assets, creados por el usuario, existen una gran cantidad de estos agrupados en unos elementos llamados Packages, creados por otros usuarios o por la empresa a cargo del motor. Para este proyecto se han utilizado tanto assets propios como creados por la empresa y por algunos usuarios que ofrecían diversos de forma gratuita y sin ningún tipo de copyright.

7.1.3 Scripts

Los scripts son los ficheros de código fuente que contiene la lógica del videojuego. En Unity estos Scripts se pueden realizar tanto en C# como en JavaScript. En ambos casos, los scripts al generarse utilizan una serie de elementos de forma automática, así como herencias varias de scripts propios del motor. En concreto el más importante es el "*MonoBehaviour",* el cual contiene las rutinas de ejecución básicas. El motor, a la hora de realizar la ejecución de estos scripts sigue un orden muy concreto dividido en varios pasos.

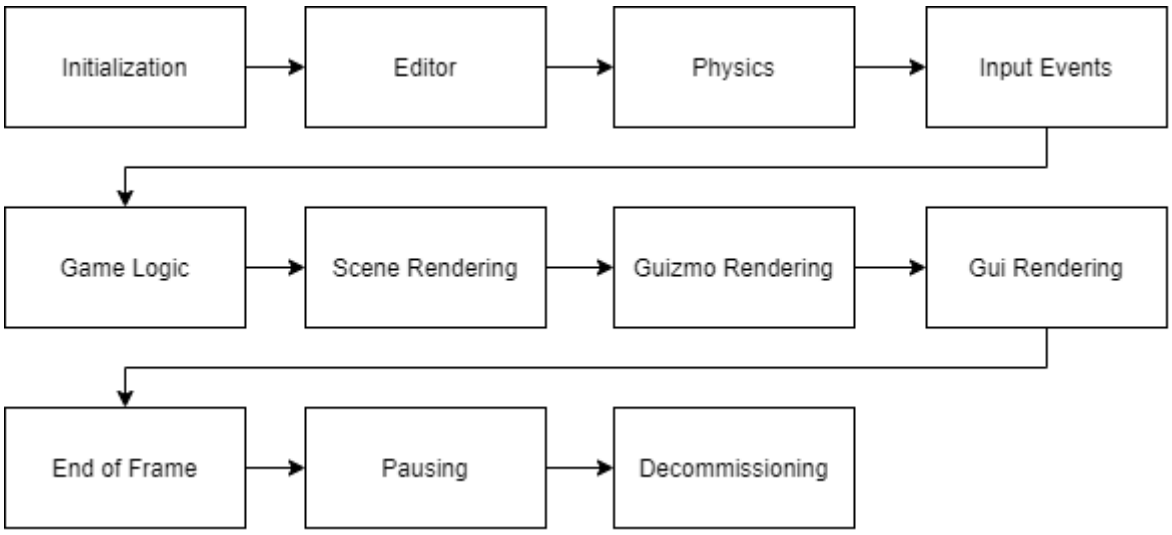


Figura 51- Scripts

Fuente: Elaboración Propia

7.1.4 Game Objects

Los GameObjects son los elementos más importantes del motor, estos elementos permiten visualizar los assets del título. Son los elementos que tienen una funcionalidad dentro del mismo, ya sea a nivel lógico, visual o sonoro. La mayor virtud de los GameObjects es que son capaces de incluir una gran cantidad de componentes.

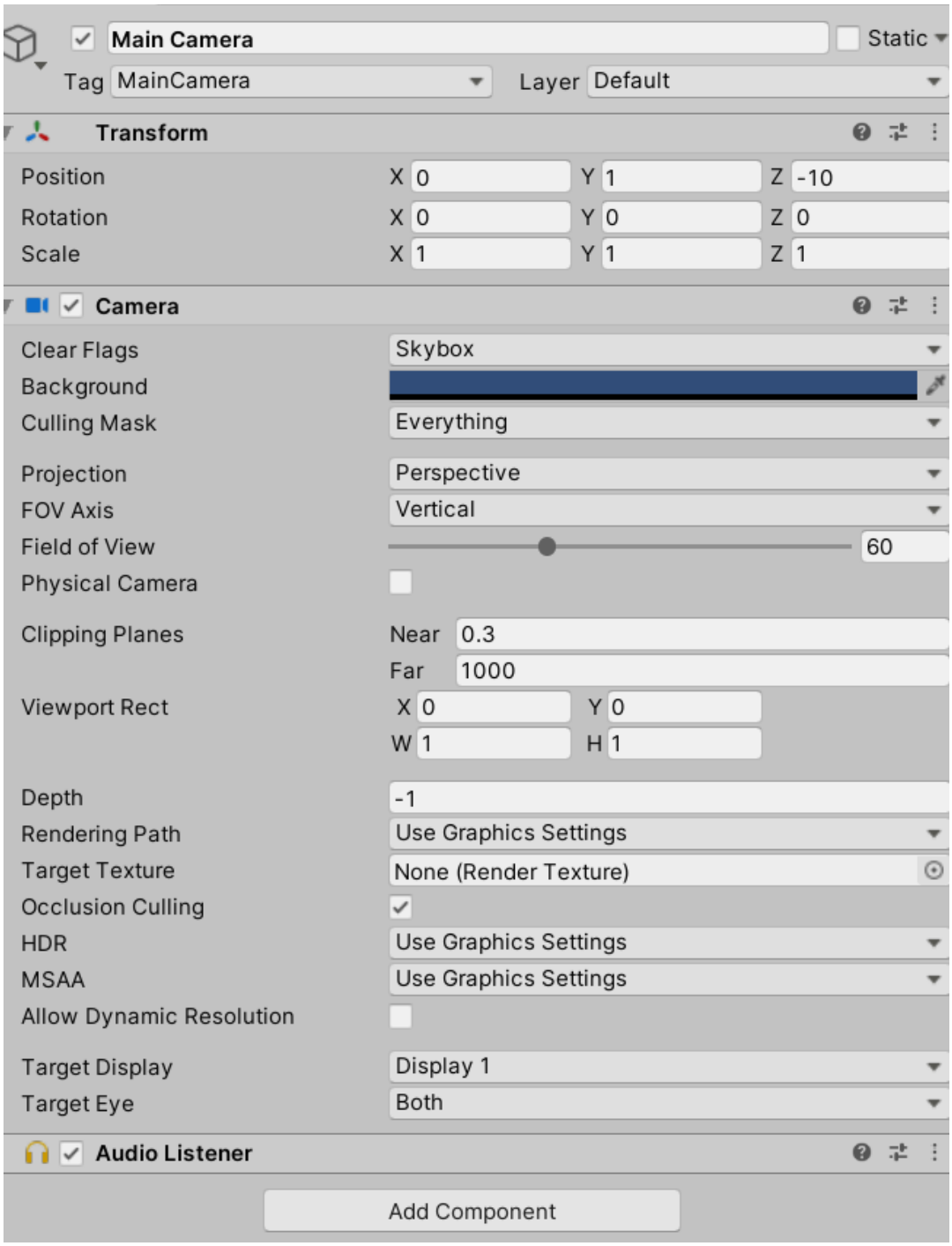
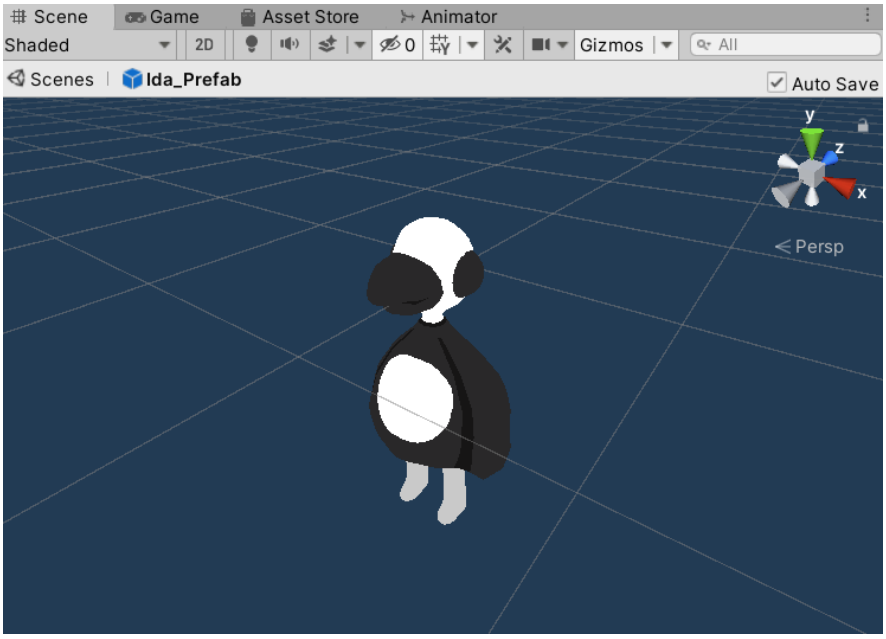


Figura 52- Game Objects

Fuente: Elaboración Propia

7.1.5 Prefabs

Los prefabs son en realidad plantillas, se utilizan cuando un GameObject aparece muchas veces en la escena. Su funcionamiento es sencillo, ya que solo tenemos que convertir el GameObject en un prefab y empezar a crear múltiples instancias de este.



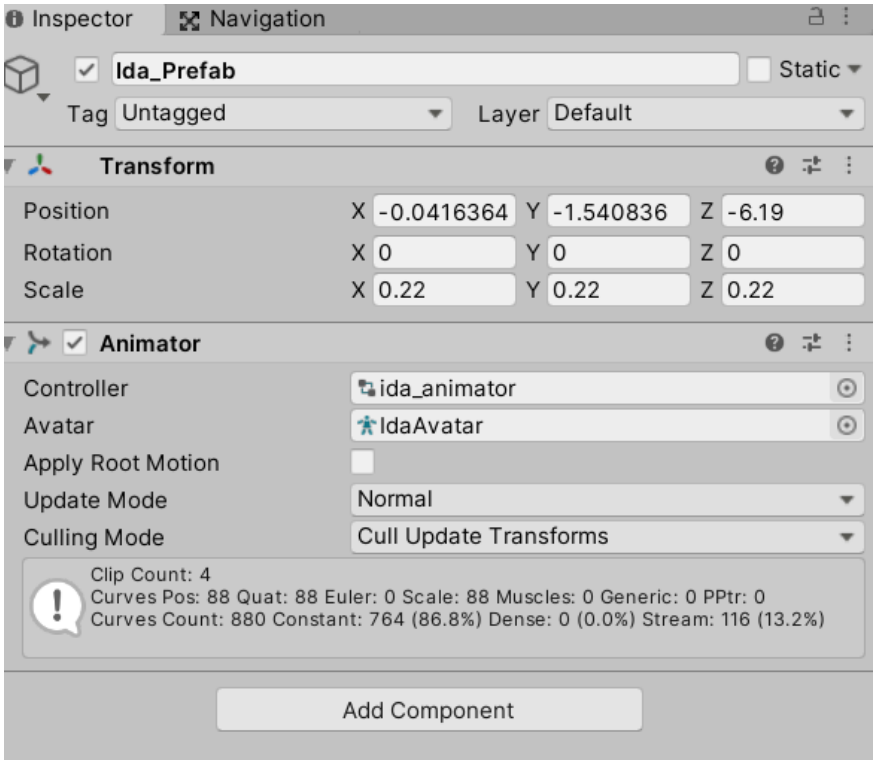


Figura 53-prefabs

Fuente: Elaboración Propia

7.1.6 Componentes

Los componentes son los elementos que dotan de diversas características a los GameObjects, estos componentes pueden ser de muchos tipos. A continuación, se explican brevemente, posteriormente se explicarán en usos concretos.

- **Cámara**: Este componente convierte a cualquier GameObject directamente en un objeto de tipo cámara, no hay límites para el número de cámaras que pueden tener una escena.

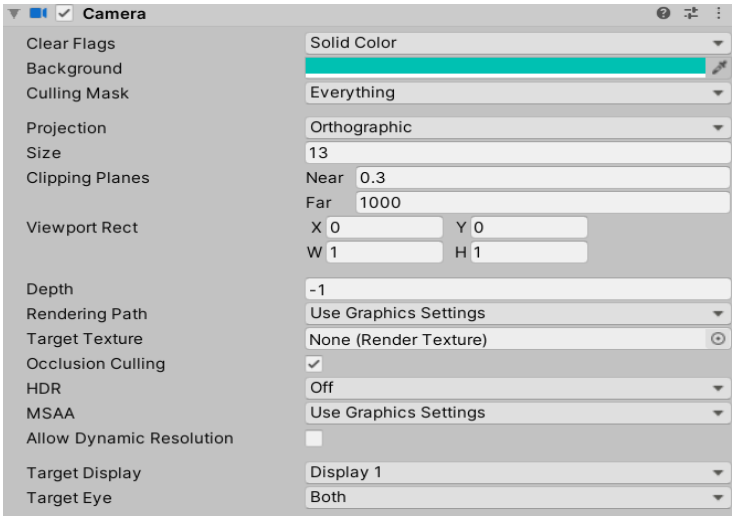


Figura 54– Componentes

Fuente: Elaboración Propia

- **Luz**: Componente que permite emitir luz. Dispone de multitud de opciones, pero en to the Heaven apenas hay trabajo luminoso.

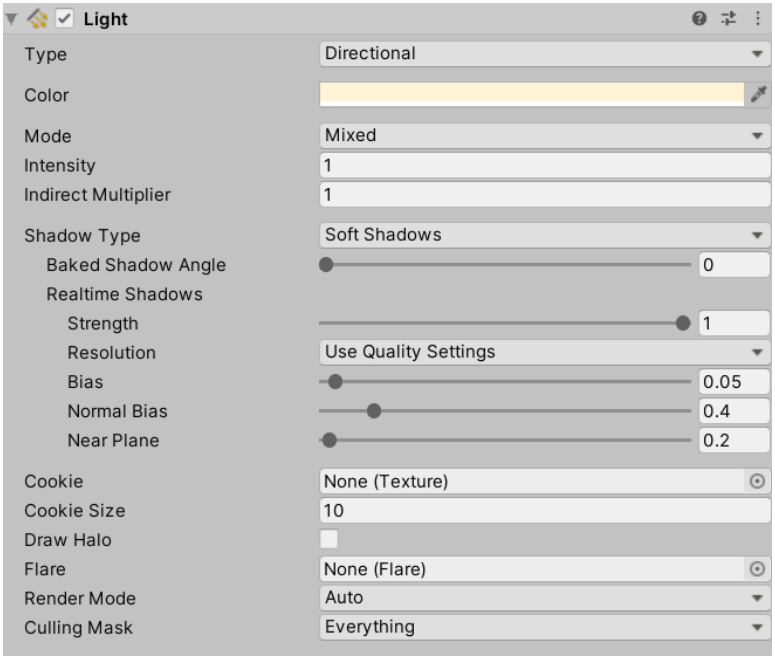


Figura 55-Componente Luz

Fuente: Elaboración Propia

- **Físicas**: Dota de comportamientos basado en reglas físicas a los objetos, los elementos más importantes son los colliders, que sirven para determinar la zona de colisión de los elementos, aunque existen muchos otros elementos.

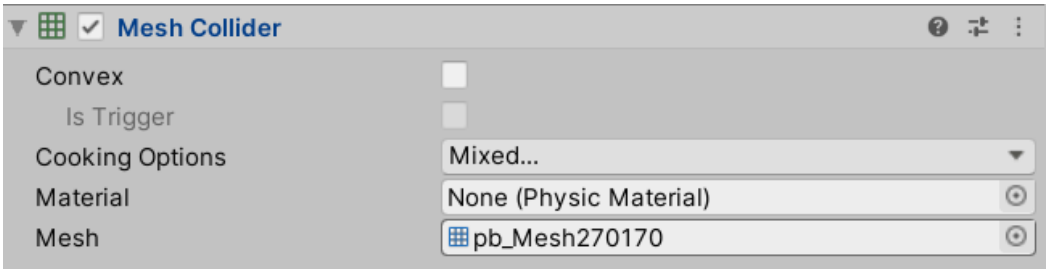


Figura 56- Collider

Fuente: https://koenig-media.raywenderlich.com/uploads/2011/08/unity14_diagram.png

- **Audio**: Convierte a los objetos en emisores de audio.

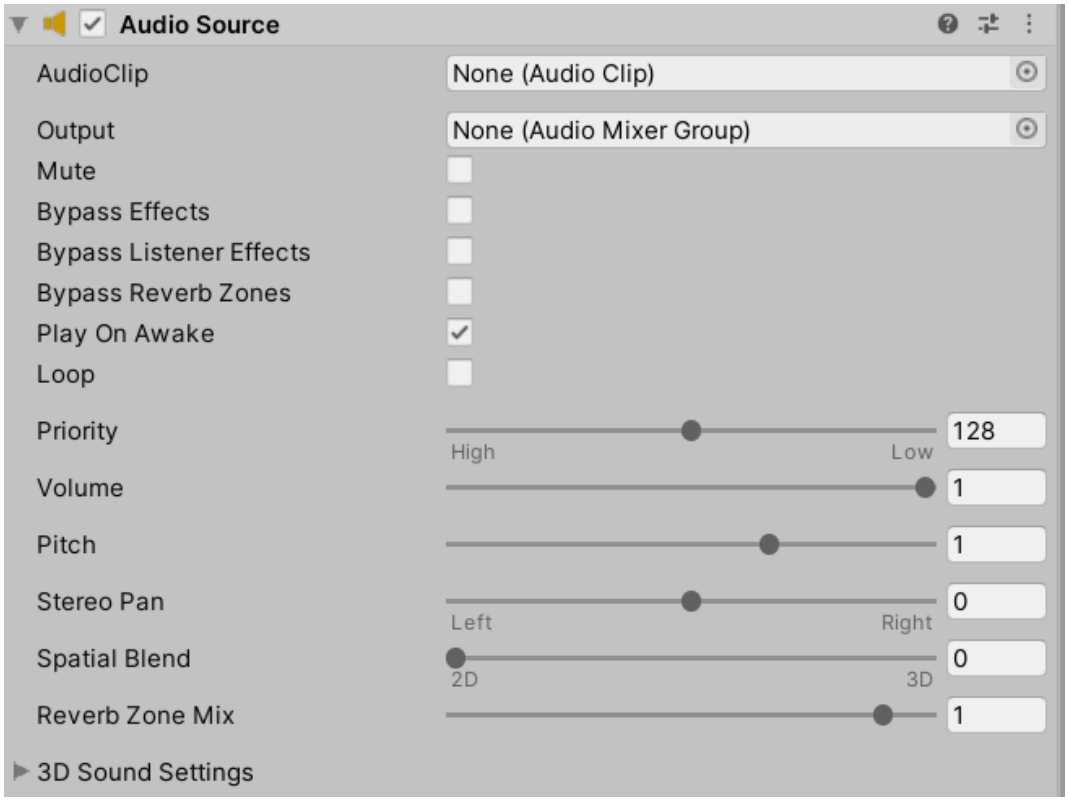


Figura 57- Audio Source

Fuente: https://koenig-media.raywenderlich.com/uploads/2011/08/unity14_diagram.png

- **Mesh**: Permite añadir a un GameObject objetos 3D más complejos. Los elementos como los materiales también se añaden en este componente.

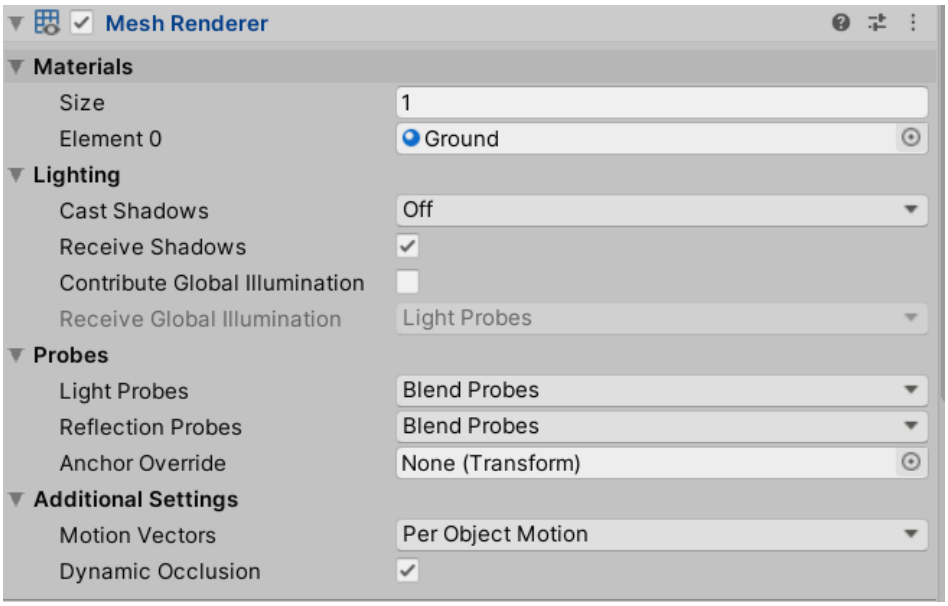


Figura 58-Mesh

Fuente: https://koenig-media.raywenderlich.com/uploads/2011/08/unity14_diagram.png

- **Animators**: Componente que permite establecer la secuencia de animaciones del personaje. La imagen inferior muestra un animator simple

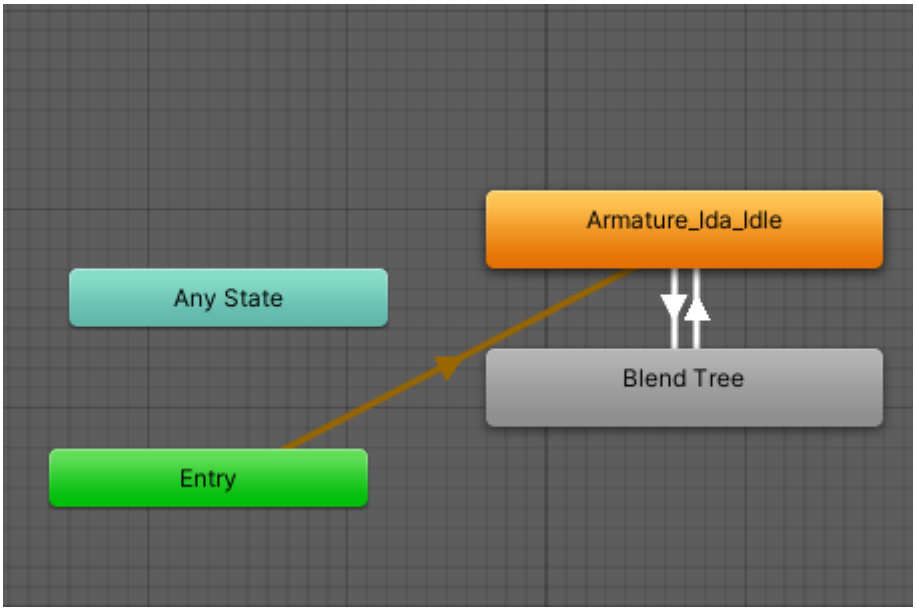


Figura 59-Animator

Fuente: https://koenig-media.raywenderlich.com/uploads/2011/08/unity14_diagram.png

- **Scripts**: Los scripts se asocian a un GameObject para darle e este algún tipo de comportamiento específico.

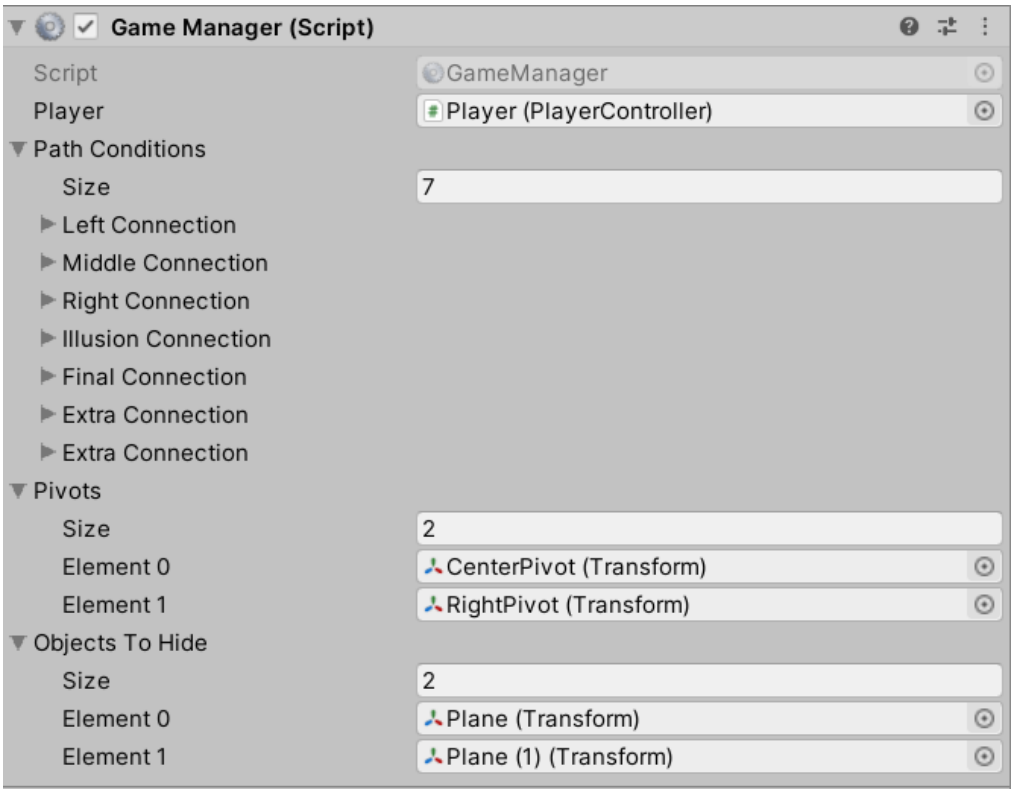


Figura 60- Script

Fuente: https://koenig-media.raywenderlich.com/uploads/2011/08/unity14_diagram.png

7.2 Bloques de Trabajo

Para realizar el desarrollo, siguiendo la planificación realizada, se han establecido 3 bloques de objetivos según su prioridad y tiempo necesario estimado para su desarrollo.

Bloque 1: Aproximadamente cuatro semanas

- Control del ciclo del juego y los eventos que suceden.
- Movimiento y Pathfinding.
- Puzle e ilusiones ópticas.

Bloque 2: Aproximadamente seis semanas

- Diseño e implementación de nuevos niveles.
- Implementación de la música.
- Efectos visuales.

Bloque 3: Aproximadamente tres semanas

- Diseño e implementación de IA básica.
- Corrección de errores.
- Implementación de Menús.
- Diseño e implementación de nuevos niveles.

7.2.1 Bloque 1

Pathfinding

El pathfinding es un concepto que hace referencia a la búsqueda del camino más corto entre dos puntos, es algo muy utilizado en la industria del videojuego actual y que es importante en To the Heaven.

El objetivo de esta técnica en el juego es controlar el movimiento del personaje para que se desplace siguiendo únicamente el camino posible. Para ello se ha realizado la siguiente implementación:

El sistema funciona con varias clases y Scripts, el primer paso es el script Walkable. Este script realiza la tarea de definir que bloques son accesibles y cuando lo son, esto es muy

importante, porque algunos puzles bloquean caminos hasta que son resueltos, su funcionamiento se explica a continuación.

Componente Walkable (Script)

En este componente se definen una serie de variables y funciones. En lo referente a las variables tenemos:

PossiblePaths*:* Lista de WalkPaths que define los caminos a los que el personaje puede moverse desde el bloque en el que se sitúa.

WalkPaths*:* Objeto que almacena las transformaciones y si está activo (bloque valido).

previousBlock*:* Transformación que almacena los datos del bloque inmediatamente anterior al bloque actual.

movingGround*:* Variable booleana que marca si el bloque en cuestión se verá afectada su posición debido a otros elementos del juego.

isButton*:* Variable booleana que marca si el bloque actual contiene un botón.

isStair*:* Variable booleana que marca si el bloque actual pertenece a unas escaleras

dontRotate*:* Variable booleana que marca si el bloque actual debe rotar.

Los métodos que implementa este componente son:

getWalkPoint: Devuelve un vector de posición con los valores del bloque.

OnDrawGizmos: Método que dibuja una serie de Gizmos de forma esférica en la escena. Los gizmos son, a grandes rasgos, elementos que nos ayudan de manera visual a identificar elementos en la escena, en este caso se utilizan para marcar los caminos accesibles desde cada bloque.

Figura 61- Gizmos

Fuente: Elaboración Propia

La segunda parte del sistema de pathfinding se encuentra en el componente PlayerController, el cual utiliza diversos valores del componente Walkable para establecer el recorrido que debe realizar hasta el bloque seleccionado, si este es válido

Componente Player Controller (Script)

En este componente se definen una serie de variables y funciones. En lo referente a las variables tenemos:

Walking*:* Booleano que indicia si el personaje está caminando o no.

Current Cube*:* Transformación del cubo actual.

ClickedCube*:* Transformación del cubo objetivo.

Indicator*:* Transformación del elemento visual que aparece al pulsar en un cubo.

FinalPath*:* Variable de tipo lista que marca el camino completo por el que tiene que pasar el personaje para llegar al objetivo.

Los métodos que implementa este componente son:

Start: Inicialización y llamada al método RayCastDown

Update: Método complejo, en primer lugar, llama a RayCastDown y realiza una serie de comprobaciones para obtener el cubo sobre el que se encuentra el personaje. Posteriormente, si se ha detectado un input por parte del usuario sobre el escenario, se realiza un pequeño render de un efecto visual que nos indica el cubo sobre el que se ha pulsado.

FindPath: Genera una serie de listas en las que se encuentran almacenados los cubos siguientes y anteriores a los que se encuentra el personaje.

ExploreCube: Trabaja los datos de las listas creadas para que en cada iteración

BuildPath: Utiliza los datos generados en el método FindPath para generar el camino definitivo hacia el cubo pulsado.

FollowPath: Con todos los datos anteriores, si es necesario, se modifican los datos de las transformaciones para un correcto movimiento en el cubo actual.

RayCastDown: Utiliza el elemento Ray de Unity para calcular la posición actual del personaje.

OnDrawGizmos: Funcionamiento similar al Gizmos del componente Walkable.

GetBlend: Obtener el valor de la variable Blend

SetBlend: Cambiar el valor de a variable Blend.

Movimiento, Animación y gestión de eventos.

El movimiento del personaje, una vez el pathfinding este operativo, se encuentra funcionando en el script Game Manager.

Componente Game Manager (Script)

Player: Componente Player Controller.

PathConditions:

- **PathConditionName:** String con su nombre.
- **Conditions:** Objeto que contiene datos de transformaciones y ángulos.
- **Paths:** Objeto que contiene, un objeto Walkable y un índice.

Pivots: Lista de Transformaciones.

ObjectstoHide: Vector de transformaciones de objectos que se van a ocultar.

SinglePath: Objeto que contiene el bloque caminable y un índice.

Los métodos que implementa este componente son:

Awake: Crea la instancia del Game Manager

Update: Método Clave, en este método se procesan los datos sobre los caminos generados y su estado, si están disponibles o no. Posteriormente, se maneja el input del jugador, para mover el jugador y los elementos interactuables del escenario, además, también se controla si el personaje está ya en movimiento para limitar los inputs y evitar los errores. Además, también se controlan la activación de ciertos elementos que activarán las rotaciones.

RotateRightPivot: Aplica una rotación a ciertos elementos.

CheckLevel: Método que controla la posición del jugador, si este llega una posición determinada, el final del nivel, finaliza este y carga el siguiente nivel.

La animación del movimiento se desarrolla conjuntamente en dos componentes que se relacionan entre sí. Un script y un animator.

Componente Player Animation (Script)

En este componente se definen una serie de variables y funciones. En lo referente a las variables tenemos:

Animator: Objeto de tipo Animator.

PlayerController: Objeto de tipo Player Controller.

Los métodos que implementa este componente son:

Update: Método que controla la activación del objeto animator en función de si el personaje se está moviendo o no.

Componente Animator (Animator)

El componente animator tiene un funcionamiento muy especial, consiste en realizar un diagrama de estados en los que se tiene que definir a que estados puede transicionar desde cualquier estado. En este caso los estados existentes son únicamente cuando el personaje está quieto o cuando está en movimiento.

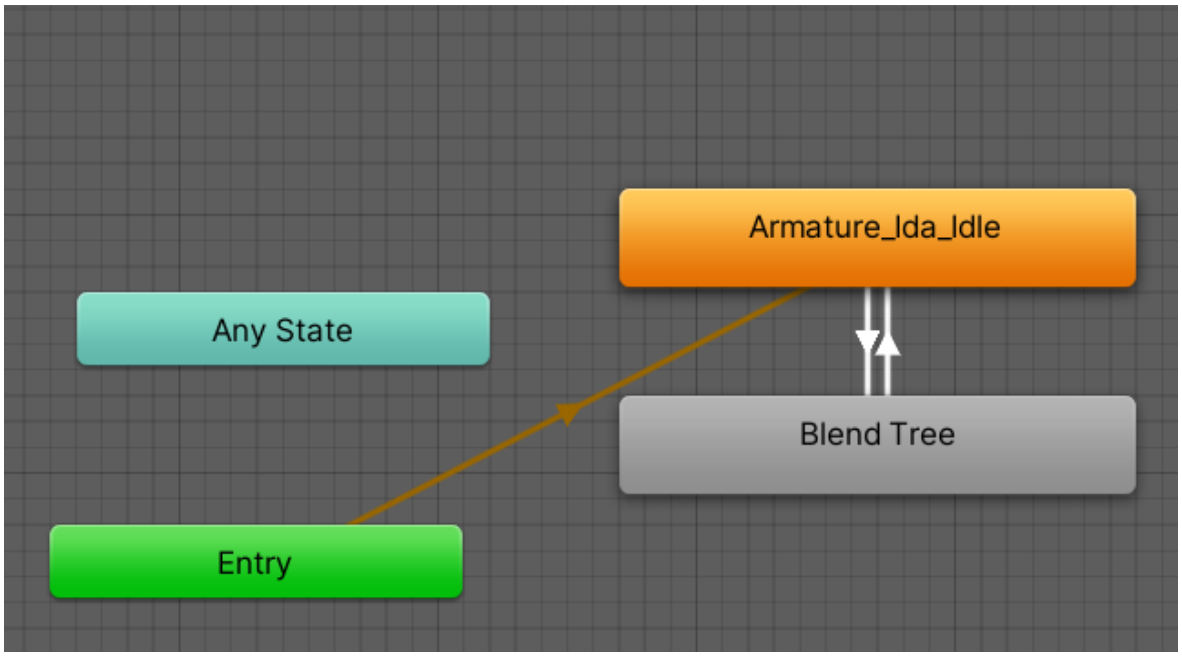


Figura 62- Animator

Fuente: Elaboración Propia

Cada estado se configura individualmente, las flechas indican los sentidos en los que puede viajar el controlador y además cada uno de ellos tiene una serie de parámetro configurables. Como la duración y el número de Frames que tiene la animación.

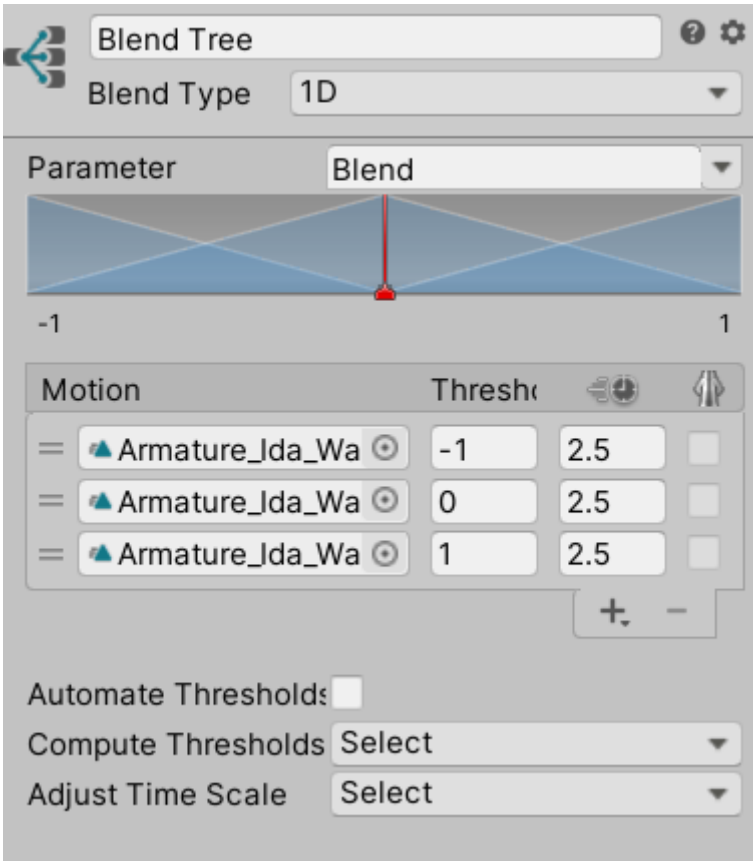


Figura 63- Animator parámetros

Fuente: Elaboración Propia

Puzle e ilusiones ópticas

Los puzles e ilusiones ópticas surgen de la combinación de varios elementos, los cuales son:

- La cámara.
- El diseño de los escenarios.
- Superficies especiales.
- Activación de elementos.

Cámara

La cámara en To the Heaven se ha configurado de una manera muy específica para resaltar algunos elementos y para realizar las ilusiones ópticas, las configuraciones más importantes son las siguientes:

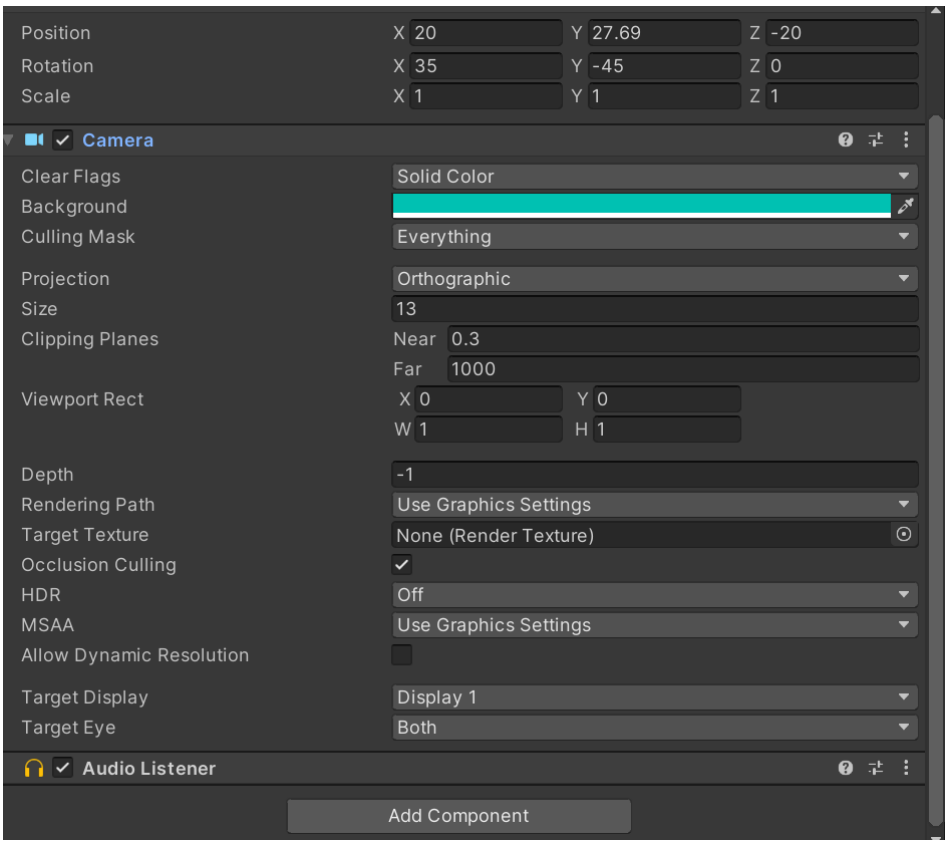


Figura 64- Configuración Cámara

Fuente: Elaboración Propia

La posición, ángulo y demás parámetros se ha configurado así después de numerosísimas pruebas siendo estos valores óptimos.

Estos valores, en conjunto con el tipo de proyección escogida, ortográfica, permite engañar al jugador y ocultar los trucos tras el diseño.

El motivo de la elección de este tipo de proyección es el siguiente, en la vista de perspectiva podemos apreciar que los objetos tienen diferentes tamaños dependiendo de las distancias en la que se encuentran uno de otro; es otras palabras; en la vista Ortogonal u Ortográfica no podemos notar la diferencia de tamaño independientemente de la distancia entre los objetos, tal cual podemos apreciar en las siguientes imágenes

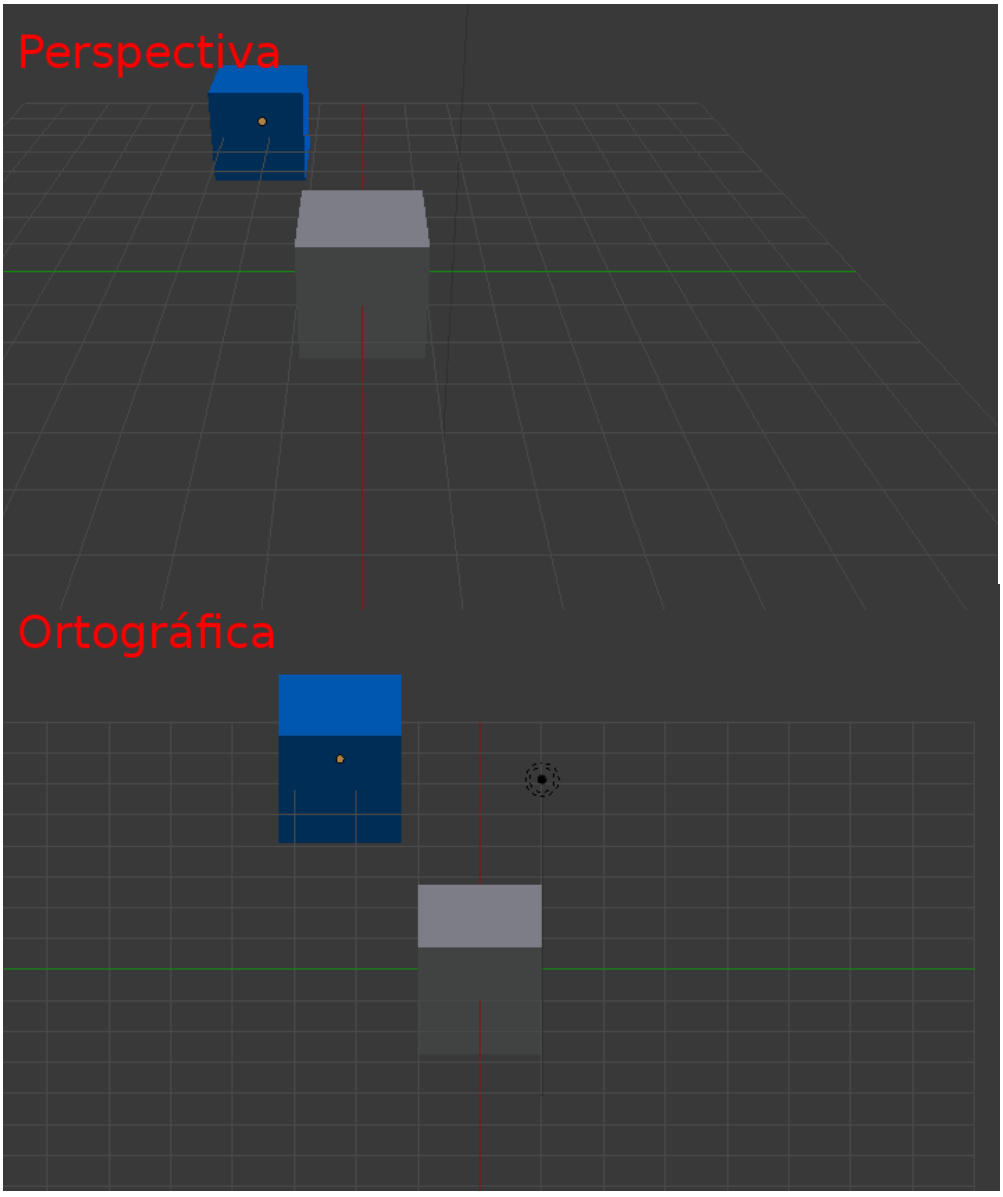


Figura 65- Perspectivas

Fuente: https://www.desarrollolibre.net/blog/blender/diferencias-entre-proyeccion-perspectiva-y-ortogonal-ortografica#.X99X_9hKjyQ

Diseño de escenarios y Activación de elementos

Para generar los puzles y para que el usuario perciba la ilusión óptica, el aspecto real del escenario, no es como lo percibe el usuario. La posición de muchos de los elementos de los niveles no están realmente donde el usuario cree que están.

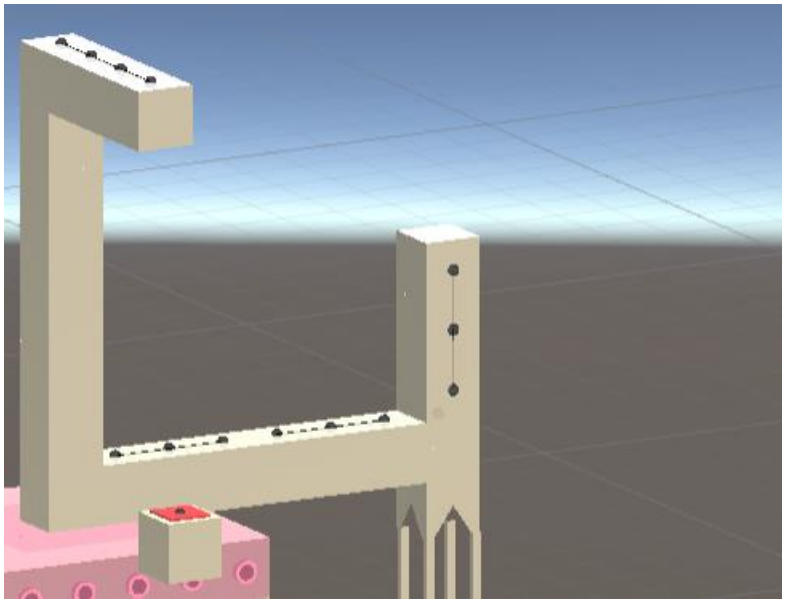


Figura 66- Engaño Visual

Fuente: Elaboración Propia

Como se puede observar en la segunda figura, varios elementos están fuera de posición, y es la cámara la que genera la ilusión. Esta posición variará también al activar elementos del escenario, como el botón rojo de la imagen.

Superficies especiales

Para poder ocultar la posición de varios elementos y que no se superpongan en la escena, en algunas superficies se han añadido planos cuyo componente de render se ha modificado para que solo se vean desde un lado.

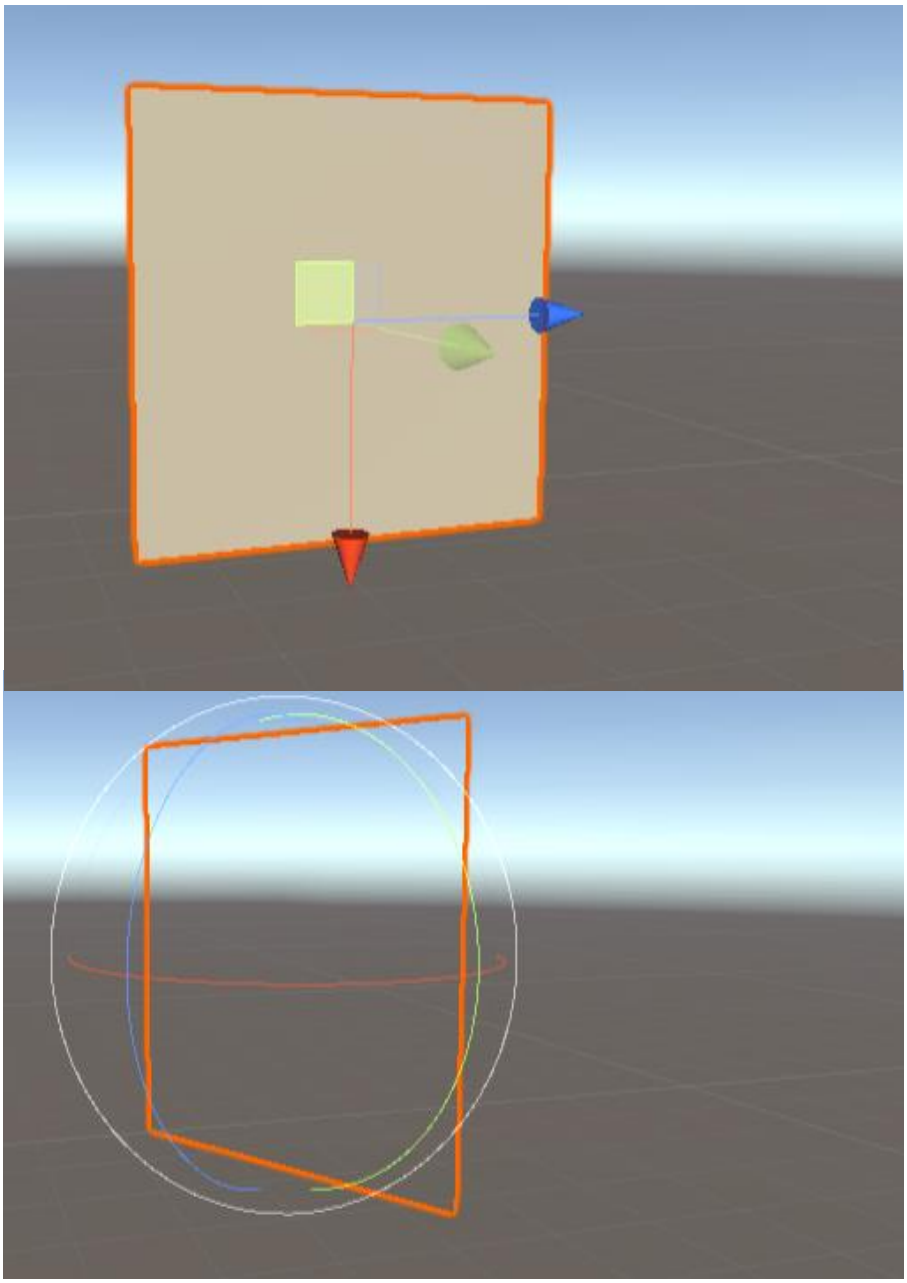


Figura 67- Efectos render

Fuente: Elaboración Propia

7.2.2 Bloque 2

Diseño e implementación de Niveles

Para este apartado, que es con diferencia el apartado del videojuego que más trabajo y horas requiere, se ha seguido una metodología concreta dividida en varios pasos.

1. Bocetar los niveles: Este paso consiste en diseña la idea principal del nivel y realizar un pequeño esquema a mano de cómo debería funcionar el nivel.
2. Modelados de los niveles en Magica Voxel.
3. Realización de los modelados en el Motor, utilizando probuilder y progrid
4. Implementación específica de un Nivel y finalización

Una vez finalizado este proceso, los scripts y componentes ya existentes se modifican para el nivel en cuestión.

Efectos Visuales

Para dotar de mejor aspecto al videojuego y enriquecerlo visualmente, se ha implementado un sistema de partículas. Este sistema de partículas funciona a través de un GameObject al que se le ha añadido un componente de particle system. Este componente es muy sencillo y consiste en configurar una serie de parámetros para generar estas partículas de acuerdo con estos parámetros. En las siguientes Figuras se muestran el resultado de este sistema y la configuración de los componentes.

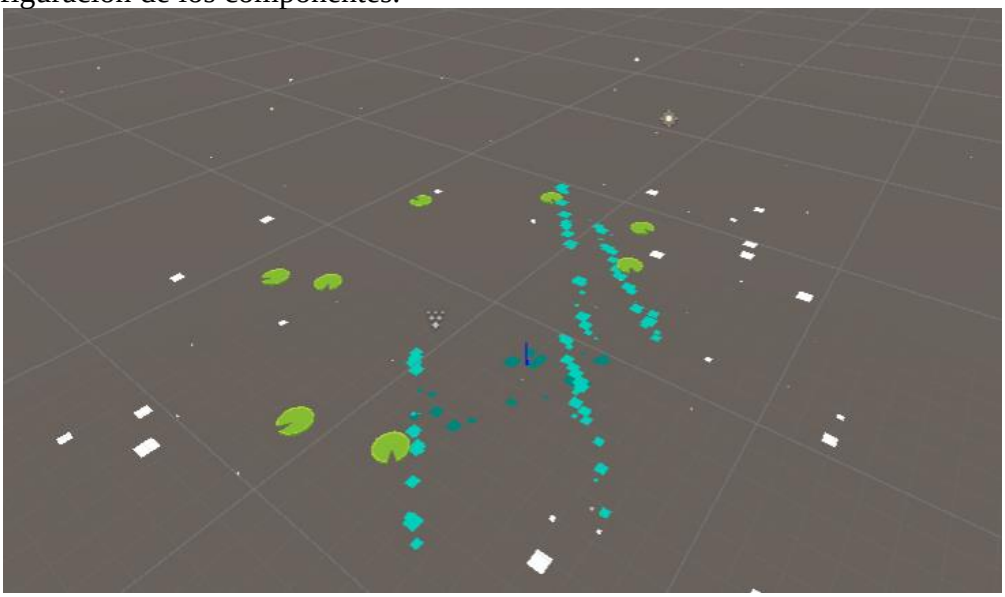


Figura 68- Párticulas

Fuente: Elaboración Propi

El resultado visto en la figura anterior surge de la siguiente configuración del sistema de partículas.

Figura 69- Partículas configuraciones

Fuente: Elaboración Propia

Cabe destacar de todos esos parámetros **la duración, el start size,** valor que indica el tamaño inicial y máximo de las partículas, **y el max particles,** número máximo de partículas generadas por sistema**.**

El siguiente aspecto relevante de las partículas es la forma, dada la estética voxel del juego

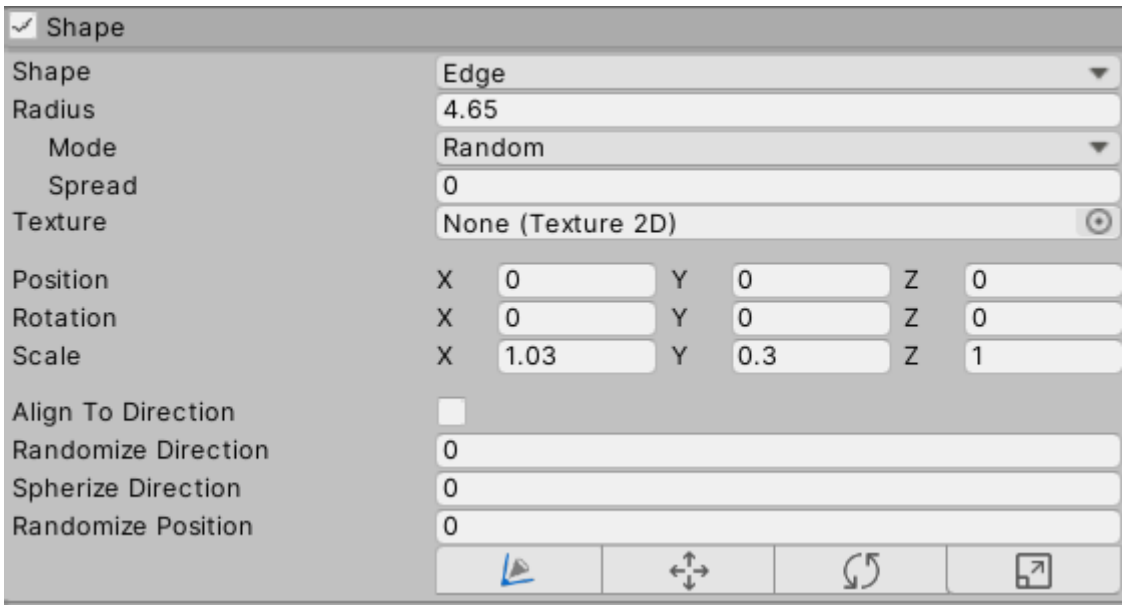


Figura 70- Partículas configuraciones

Fuente: Elaboración Propia

Para poder visualizar el sistema de partículas desde la escena y poder realizar configuraciones rápidas y simulaciones utilizamos la ventana del particle system.

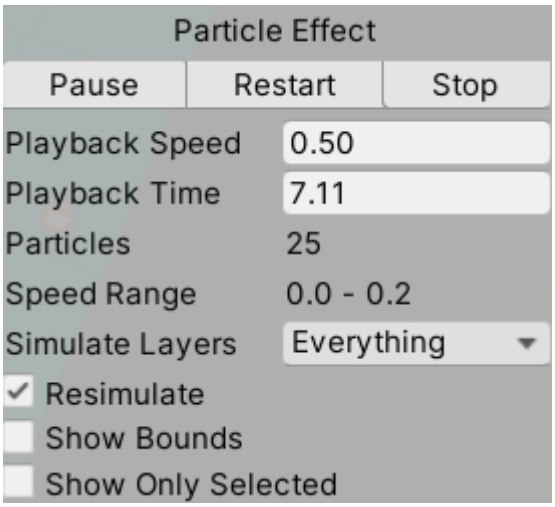


Figura 71-Partículas configuraciones

Fuente: Elaboración Propi

Audio.

El sistema de audio en Unity funciona con la unión de dos componentes, el primero de ellos es **el audio listener**. El **audio listener** permite poder escuchar sonidos en el juego, este componente se añade por defecto a al GameObject que hace de cámara.

Figura 72- Audio

Fuente: Elaboración Propia

El otro elemento es el **audio source,** este componente hace de emisor de sonido, además, se deben indicar diversos parámetros.

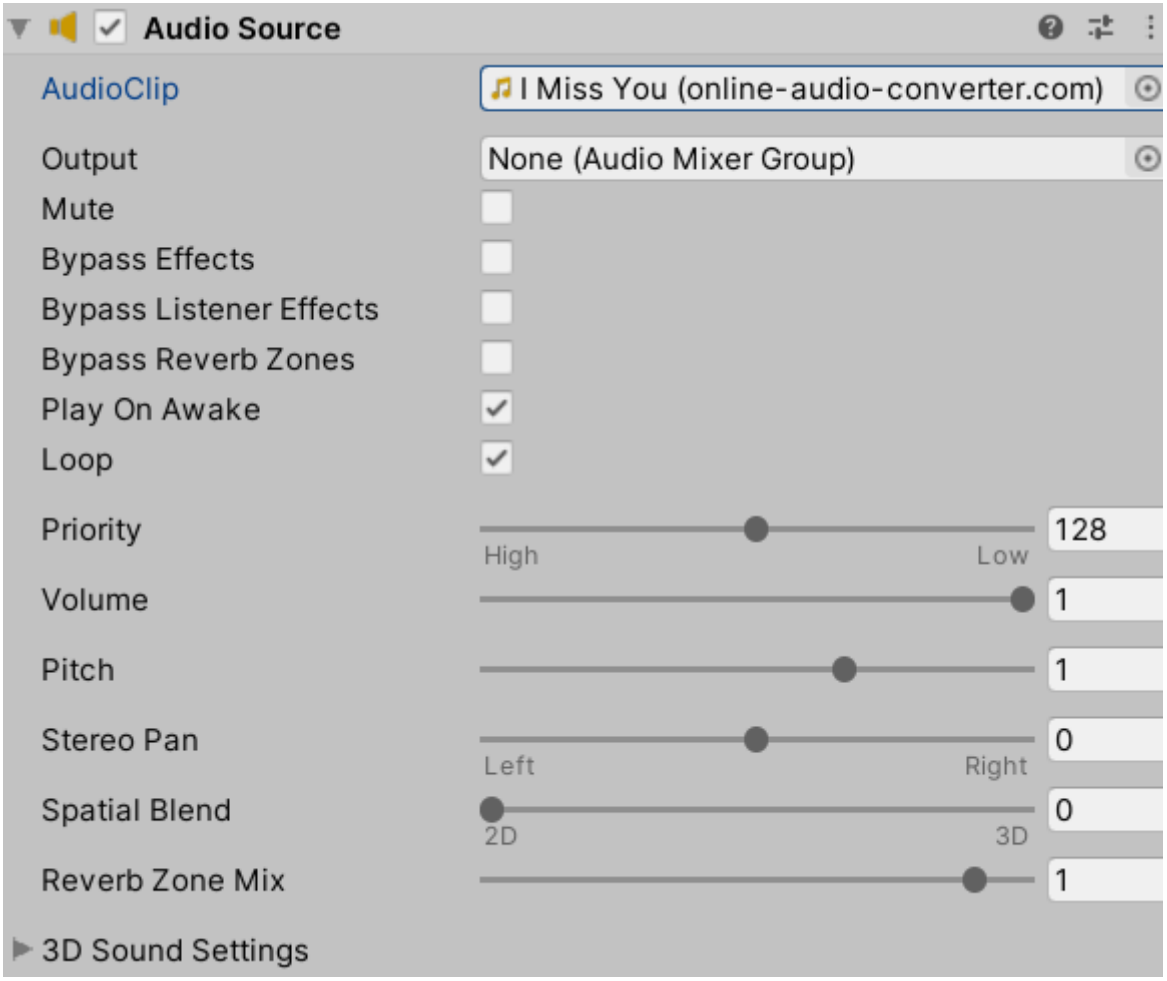


Figura 73- Audio

Fuente: Elaboración Propio

Para este tipo de niveles, donde no se puede saber cuánto tiempo va a tardar el jugador en resolver los minipuzles es importante destacar la opción de **Loop**, la cual hará que la música vuelva a sonar cuando termine.

La opción de **play on awake** hará que la música comience a sonar cuando se inicie el nivel.

7.2.3 Bloque 3

Siguiendo con la planificación realizada, en este bloque se debería continuar con los siguientes puntos.

- Diseño e implementación de IA básica.
- Corrección de errores.
- Implementación de Menús.
- Diseño e implementación de Niveles.

Debido a situaciones ajenas al proyecto se han tenido que aplicar varios planes de contingencia, causando que este bloque de desarrollo se tenga que suspender, dejando sin implementar el menú y varios niveles. A continuación, en el apartado de conclusiones se explican los motivos de esto entre otras cosas.

8. Conclusiones

En este apartado hablaremos sobre las conclusiones, tanto a nivel del proyecto, analizando la consecución de objetivos, como a nivel personal, dando un punto de vista personal final.

8.1 Análisis de Objetivos

En el apartado de justificación y objetivos se planteó una lista de objetivos.

Relacionado con Unity:

Objetivo	%	Motivos
Programación en C#.	70	Se ha aprendido mucho sobre programación en C#, su sintaxis y la forma de trabajar, aunque hubiera sido ideal poder profundizar más.
Flujo de trabajo en Unity.	50	Se ha aprendido mucho sobre como trabajar con esta herramienta, el problema ha sido que no te planteó bien al inicio y esto provocó pérdida de tiempo.
Programación Móvil	60	Se ha aprendido a realizar ciertas optimizaciones para que el juego siga manteniendo una buena calidad sin comprometer el rendimiento.
Funcionalidades avanzadas y Scripting	50	Se han realizado múltiples scripts para diferentes comportamientos, pero diversos elementos no se han podido implementar.
Programación eficiente	70	Se ha aprendido a realizar ciertas optimizaciones para que el juego siga manteniendo una buena calidad sin comprometer el rendimiento.

Tabla 8- Unity

Fuente: Elaboración Propio

Objetivo	%	Motivos
Fases de Desarrollo	85	Se ha realizado un desarrollo de software completo, realizando todas sus fases, desde la concepción de una idea hasta su realización(parcial) pasando por todos los análisis previos.
Documentación desarrollo	90	Se ha documentado el desarrollo al completo de la forma planificada casi en su totalidad, a falta de algunos detalles.
Investigación	100	Se ha realizado una gran investigación tanto de forma previa como durante el desarrollo.
Estimación de costes	100	Se ha realizado una estimación de costes completa y detallada.
Modelo de Negocio	100	Se ha realizado un análisis de los modelos de negocio actuales.
Planificación	100	Se ha realizado una planificación inicial concreta, además de varias replanificaciones.
Metodologías.	100	Se realizó un análisis de metodologías para decidir entre ágiles y tradicionales.

Tabla 9- Objetivos Generales

Fuente: Elaboración Propio

8.1 Problemas encontrados

Durante el desarrollo del proyecto han ocurrido, como era de esperar, una gran cantidad de problemas, a continuación, se muestra una tabla resumen con los problemas surgidos y como se atajaron o como se intentó minimizar sus efectos.

Problema Surgido	**Contingencia Realizada**	**Afectación final al proyecto**
Problemas Familiares	Se han tenido que descartar muchos elementos para poder llegar a la fecha con un producto mínimo viable.	Crítica
Estrés Elevado	Debido a la acumulación de diversas situaciones, se tuvo que suspender totalmente el desarrollo durante varias semanas. Lo cual sumado al problema anterior provocó grandes replanificaciones	Grave
Cálculo erróneo de la dimensión del proyecto	Priorizar las tareas críticas y establecer prioridades,	Moderada
Rotura pc	Se disponía de un segundo equipo que permitía seguir trabajando en otras tareas menos exigentes técnicamente.	Leve
Tareas mal fraccionadas	Realizar diversas replanificaciones	Leve

Enfermedad	Modificar algunas iteraciones, cambiando la prioridad de ciertas tareas.	Leve

Tabla 10- Riesgos

Fuente: Elaboración Propio

8.1 Conclusiones Personales

Este trabajo ha sido una experiencia personal realmente interesante, he conseguido realizar en buena parte los objetivos concretos y he aprendido muchísimo, es más, diría que he aprendido más de las cosas que no he conseguido realizar que de las que sí. Aunque me hubiera gustado realizar todo lo que me propuse, esto refleja lo complicado que puede ser realizar un proyecto y el como las situaciones ajenas al mismo pueden afectar al mismo.

He conseguido plasmar muchos de los conocimientos adquiridos durante la titulación en este proyecto, la cual no ha sido fácil y ha requerido mucho esfuerzo, igual que este trabajo, pero al final, la experiencia ha valido la pena, independientemente del resultado final. Pues con lo que he aprendido estoy convencido de que ahora haría un proyecto como este mucho mejor y más rápido.

Aunque finalmente a nivel de estudio me han surgido otras oportunidades no relacionadas con los videojuegos, esta experiencia ha ratificado lo mucho me gustan los videojuegos y la cantidad de gente necesaria para pode realizar un título de gran calibre.

Esto marca el fin de una etapa de mi vida muy importante, espero que siga aprendiendo y pueda demostrar todo lo que he aprendido.

9. Bibliografía

Antonieza Kuz, M. F. (2018). *Comprendiendo la Aplicabilidad de Scrum en el Aula: Herramientas y Ejemplos.* La Plata. Obtenido de http://portal.amelica.org/ameli/jatsRepo/24/2414011/html/

Atari. (2000). *Tetris History.* Obtenido de http://www.atarihq.com/tsr/special/tetrishist.html

Atomix, S. (2013). *LO COMPLEJO Y DIVERTIDO: SOBRE LOS PUZZLES.* Obtenido de https://atomix.vg/lo-complejo-y-divertido-sobre-los-puzzles/

BOOST. (2019). *What is Price to Win?* Obtenido de https://boostllc.net/what-is-price-to-win/

Calveley, L. (2015). *Puzles: tipos, metodologías y recomendaciones.* Obtenido de http://aev.org.es/puzles-tipos-metodologias-recomendaciones/

Game Digits. (2017). *Dream Machine.* Obtenido de https://www.gamedigits.co.uk/

Gdp Master. (Unknown). *La Ley De Parkinson.* Obtenido de http://www.gestiondeproyectos-master.com/la-ley-de-parkinson/

IPYME. (Unknown). *Herramienta DAFO.* Obtenido de https://dafo.ipyme.org/Home

Julían Pérez Porto, A. G. (2010). *Definición de Videojuego.* Obtenido de https://definicion.de/videojuego/

Magni, M. (2016). *Mekorama.* Obtenido de https://www.mekorama.com/

Ruela, U. (2017). *¿Qué es un motor de videojuegos (game engine)?* Obtenido de https://codingornot.com/que-es-un-motor-de-videojuegos-game-engine

Ustwo Games. (2014). *Monument Valley.* Obtenido de https://www.ustwogames.co.uk/